AD-A259 210

AFIT/GE/92-D-15

HIGH RANGE RESOLUTION RADAR TARGET IDENTIFICATION USING THE PRONY MODEL AND HIDDEN MARKOV MODELS

THESIS

Mark R. DeWitt
Captain, USAF

AFIT/GE/92-D-15

DTIC
ELECTE
JAN 11 1993
S B D

93-00099

Approved for public release; distribution unlimited

AFIT/GE/92-D-15

HIGH RANGE RESOLUTION RADAR TARGET IDENTIFICATION USING THE PRONY MODEL AND HIDDEN MARKOV MODELS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

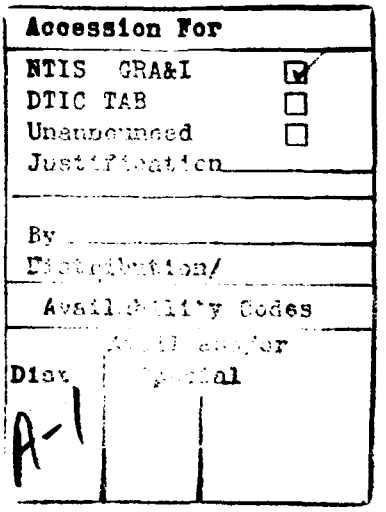
Requirements for the Degree of

Master of Science in Electrical Engineering

Mark R. DeWitt, B.S.E.E

Captain, USAF

December, 1992



Approved for public release; distribution unlimited

DTIC QUALITY INSPECTED 5

Acknowledgments

This thesis is dedicated to my wife, Rita and my children; Robin, Jonathan and Cassie. Throughout the last 18 months they have been there for me even when I wasn't always there for them.

I would like to thank my advisor, Captain Dennis Ruck, for giving me the independence to try my ideas while insuring that I wasn't straying too far from the objective. I would also like to thank Major Steve Rogers for the initial drive to pursue this research area and his keen and objective reviews as the project progressed. Captain Josseph Sacchini was instrumental by supplying the MatLab code to do the Prony Model parameter estimation and Lieutenant Colonel Edmand Libby provided valuable insight as to the overall problem. Finally, I would like to thank my colleagues, Captain James Geurts and Captain Thomas Andrews for their friendship and good humor which have made this project more enjoyable.

Mark R. DeWitt

Table of Contents

List of Figures

List of Tables

Abstract

Fully polarized Xpatch signatures are transformed to two left circularly polarized signals. These two signals are then filtered by a linear FM pulse compression ('chirp') transfer function, corrupted by AWGN, and filtered by a filter matched to the 'chirp' transfer function. The bandwidth of the 'chirp' radar is about 750 MHz. Range profile feature extraction is performed using the TLS Prony Model parameter estimation technique developed at Ohio State University. Using the Prony Model, each scattering center is described by a polarization ellipse, relative energy, frequency response, and range. This representation of the target is vector quantized using a K-means clustering algorithm. Sequences of vector quantized scattering centers as well as sequences of vector quantized range profiles are used to synthesize target specific Hidden Markov Models (HMM's). The identification decision is made by determining which HMM has the highest probability of generating the unknown sequence. The data consist of synthesized Xpatch signatures of two targets which have been difficult to separate with other RTI algorithms. The RTI algorithm developed for this thesis is clearly able to separate these two targets over a 10 by 10 degree (1 degree granularity) aspect angle window off the nose for SNRs as low as 0 dB. The classification rate is 100 % for SNRs of 5 – 20 dB, 95 % for a SNR of 0 dB and it drops rapidly for SNRs lower than 0 dB.

HIGH RANGE RESOLUTION RADAR TARGET IDENTIFICATION USING THE PRONY MODEL AND HIDDEN MARKOV MODELS

I. INTRODUCTION

1.1 Background

In air-to-air engagements between high-performance fighter aircraft, positive and timely identification of targets is critical. In the absence of a cooperative identification system and less than adequate intelligence, target identification must be deduced from backscattered electromagnetic energy (30). Targets without cooperative identification systems, such as Identification Friend or Foe (IFF) transponders, are called non-cooperative targets.

The backscattered or emitted electromagnetic energy from a non-cooperative target of interest can be detected by a number of systems depending on the characteristics of the signal. If the target is within visual range, the backscattered electromagnetic energy (light in this case) can be sensed by the pilot's eyes and processed by the most powerful identification system known – the human visual system. However, if the target is at a relatively long range, the backscattered electromagnetic energy must be measured electronically. Radar is well suited for this task. The target is illuminated with a well defined signal and the backscattered electromagnetic energy (radar return) is then detected by a receiver which is optimized to process the radar return. Non-cooperative target recognition (NCTR) with radar is designated Radar Target Identification (RTI).

Historically, radar has been used to estimate the relative range, direction, and velocity of targets of interest (26). Although these parameters give significant information about the target, they cannot – by themselves – be used to identify the object. Unlike the conventional radars which are used to estimate relative position and velocity, High Range Resolution (HRR) radar is specifically designed to resolve the target in range along the radar-target vector. Range resolution is defined as the minimum distance between two point targets at which they can discerned as two distinct targets (26). In other

words, if the range resolution is one meter; point targets separated by less than a meter will appear as one target. In general, the range profile is created by dividing the return into range bins the size of the range resolution of the radar.

As stated previously, the radar illuminates the target and receives the backscattered signal to yield the range profile, location, and velocity of the target. The range profile is sent to a feature extraction process which outputs a set of descriptive parameters which concisely describe the target. The approximate target/radar aspect angle (the direction at which the target is being illuminated) is estimated from the location and velocity of the target via the tracking process. Finally, the decision process chooses the target identification based on the range profile feature set and knowledge of the target/radar aspect angle.

1.2 Problem Statement

This thesis will develop a reliable RTI algorithm which uses a real aperture (as opposed to synthetic aperture) HRR radar as a sensor. This RTI algorithm will be designed to perform on-board identification during air-to-air engagements between fighter aircraft as depicted by Figure 1.1. For realistic operational scenarios, the RTI algorithm must perform the identification process with a minimum number of measurements for near-real-time target identification.

1.3 Summary of Current Knowledge

Research in the area of Radar Target Identification (RTI) spans a broad range of radar types and classification algorithms. Libby presents a comprehensive summary of the start-of-art of RTI (17). What follows is an overview of the research areas which are most applicable to RTI algorithms developed for this thesis.

1.3.1 Variability of HRR Radar Profiles and Some Implications for Target Recognition Cohen (4) provides some valuable insight as to the general nature of the narrow band radar cross section (RCS) of complex targets such as aircraft and how the RCS relates to the wideband HRR radar range profiles of these targets. To quote Skolnik, "the RCS of a target is the (fictional) area intercepting that amount of power which, when scattered equally in all directions, produces an echo at the radar equal to that from the target (26)." That is to say, if a target reflects the same energy back to the radar as a perfectly

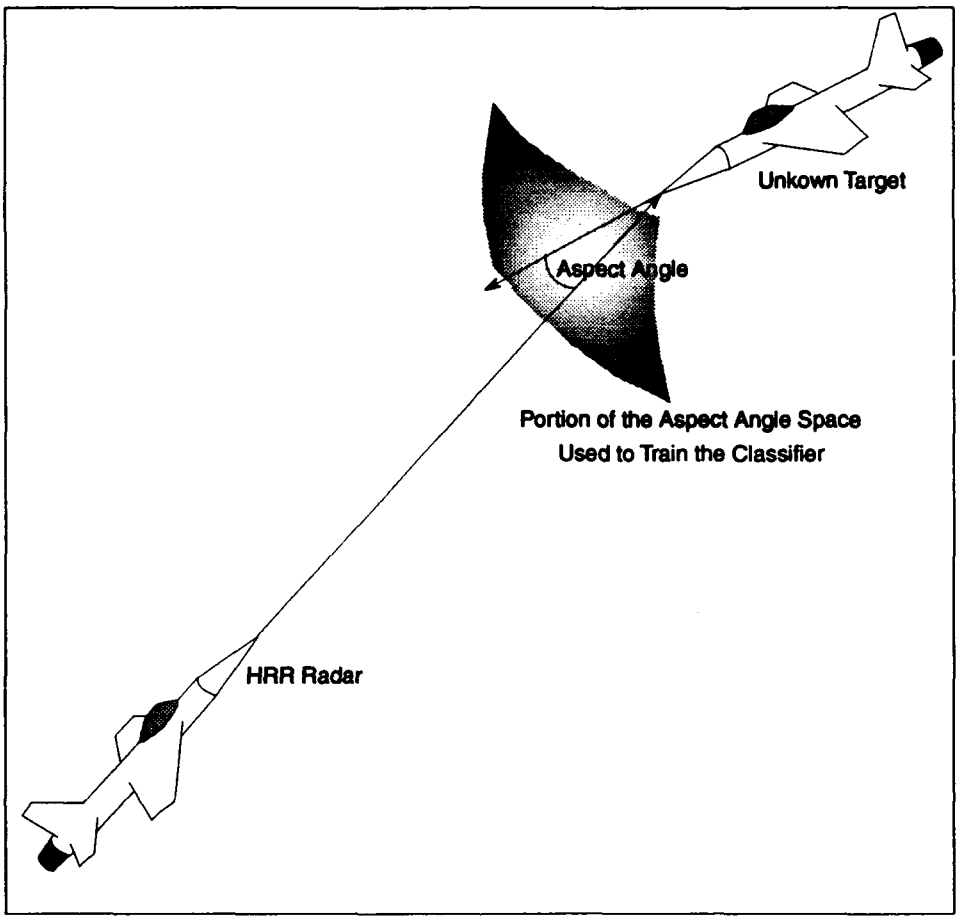


Figure 1.1. Operational Scenario

conducting sphere (which scatters isotropically) whose cross section is 1 m^2, the RCS of that target is 1 m^2.

The narrowband RCS of an aircraft or any other complex target is extremely dependent on aspect angle and therefore, cannot be reported as a single quantity. Thus, RCS is usually reported for a single frequency in a polar plot as a function aspect angle (4) (26). If the RCS is modeled as a random process, it is not uncommon to find that it can completely decorrelate every 0.2° change in aspect angle. Because of this sensitivity to aspect angle, Cohen states that RTI algorithms may require inordinate amounts of both data for training and memory for implementation (4).

Cohen showed that the the energy contained within the range profile changes with the same variability as the narrowband RCS, but general shape of the wideband range profile is less susceptible to changes in aspect than the RCS. However, the shape of the range profile is still very sensitive to changes in aspect angle. Therefore, it may be advantageous, from an RTI perspective, to normalize the energy of the range profiles to prevent their amplitudes from changing as the RCS changes. However, the loss of information which is contained in the absolute RCS and how it changes with aspect angle may negate the net improvement gained by this normalization (4).

1.3.2 Recognition of Targets Using Sequences of Range Profiles and Knowledge of Aspect Angle Libby, who is currently conducting RTI research at AFIT, has shown that the aspect angle of an aircraft in stable flight can be estimated to a high degree using the information available from a tracking system (16). In this light, he suggests that if the aspect angle is known, target identification will be improved if the classifier excludes possible matches to targets whose templates are known to be outside the tolerance of the aspect angle estimate. More importantly, Libby also states that if the classifier makes use of the temporal relationships between range profiles in a sequence, coupled with the knowledge of the aspect angle of the target during that sequence, the classification performance will be better than identification based on individual range profiles treated as independent events. Also, Libby suggests that Hidden Markov Models, which have been used for speech processing, may be useful for processing these temporally related range profile sequences (17).

1.3.3 General Dynamics RTI Research General Dynamics (GD) conducted RTI research using HRR under contract with the USAF's Wright Laboratory (1) (28). The algorithm developed by GD was based on a "slide distance metric" between range profiles from a known target and range profiles from an unknown target. Range profiles were gathered for an actual target at discrete angles and processed to extract the significant peaks along the radar/target vector. This reduced data set consisted of the most prominent peaks and was stored as a target template from which the unknown range profiles were matched. The "slide distance metric" was calculated by aligning the two range sweeps in each of four different ways – aligning the first peak of the unknown peak with the first peak of the template sweep, aligning the second peak of the unknown sweep with the second peak of the template sweep, and so on through a fourth peak-to-fourth peak match. This peak-to-peak comparison was performed for all the stored target templates and the target identification corresponded to the minimum slide distance.

Depending on the complexity of the target, several templates at different aspect angles must be available for each target.

1.3.4 Target Identification Using Polarization-Diverse Features Chamberlain used the fully polarized signatures to generate the transient polarization response (TPR) of the target (2). The TPR is generated by illuminating the target with an impulse of a circularly polarized wave. As the wave interacts with each scattering center on the target, each scattering center reflects a wave with a polarization that is determined by the polarimetric characteristics of that scattering center. The individual scattering centers were estimated nonparametrically using the Inverse Fast Fourier Transform (IFFT) of the discrete or stepped frequency response of the target. Chamberlain found that the polarization derived from the TPR related well with the shape and orientation of the major scattering centers distributed in the downrange profile of the target. The TPR features of five commercial aircraft range profiles were classified using K-nearest neighbor (KNN) technique. At a single aspect angle, the correct target identification was made 97% of the time for signal to noise ratios (SNR) as low 0 dB.

1.3.5 High Resolution Exponential Modeling of Fully Polarized Radar Returns In Chamberlain's work, nonparametric techniques for extracting the scattering centers was developed. Sacchini and Steedly (25) (27) developed a parametric method to locate scattering centers which was based on the Prony Model (23). Like the research of Chamberlain, each scattering center is characterized by a polarization ellipse which corresponds to the back scattered polarization from a circularly polarized wave. Specifically, the IFFT method and the Prony method were compared (25). Except for computational cost, the performance of the Prony model was superior to the conventional IFFT techniques. In fact, the Prony model estimated the polarization ellipses of the dominant scattering centers of various aircraft with reasonable accuracy at 0 dB SNR.

1.3.6 Automatic Target Identification Based on Models of Neural Networks There are a number examples where neural networks have been used for RTI. Farhat (8) trained a neural network with sinograms of scale models of a B-52, AWACS, and the Space Shuttle. A sinogram is a Cartesian plot of the measured relative range of scattering centers on the object versus aspect angle. Farhat found that the neural network was able to correctly identify the target with as little as 10% of the sinogram. Although the findings are significant, they may be of limited value because the sinogram used for training and

identification was generated by targets rotating smoothly through a range of azimuth angles at a constant elevation angle about the plane of the wings.

1.3.7 Hidden Markov Models Traditionally, Hidden Markov Models (HMM's) have been used for isolated word recognition (IWR) (11) (12) (13) (15) (20) (21) (22) . For this purpose, words to be identified are modeled using a HMM – one HMM per word in IWR applications– and the correct word is then chosen by determining which model had the highest probability of generating the unknown word. HMM's have been used for IWR with excellent results when compared to other techniques (12) (21) (22) . It appears that HMM's, and in particular left-right HMM's, are well suited for HRR RTI because the temporal relationship of scattering centers are inherently represented by the HMM state transitions (22).

1.4 Scope

Previous work will be drawn upon for all phases of the RTI process development. The basic theory of HRR radar has been developed for over 30 years (5) and equations derived from this knowledge base will be implemented in software. However, no attempt will be made to emulate any specific HRR radar that has been implemented in hardware. Also, the High Resolution Exponential Modeling of stepped frequency range profiles developed by Sacchini and Steedly (25) (27) will be be used for feature extraction with little modification. Therefore, the primary focus of this research will be on the actual classification algorithms. The RTI system developed for this thesis will be designed to identify two fighter class aircraft over the limited aspect angle window depicted by Figure 1.1.

1.5 Assumptions

The HRR RTI problem is further defined and scaled by the following assumptions:

- The tracking system can estimate the aspect angle to within a 10° by 10° azimuth and elevation window in the aspect angle space as shown in Figure 1.1 (16).
- The range profile is corrupted by additive white Gaussian noise and is free of other types of noise associated with clutter (26).

- The operating frequency of the radar is approximately 10 GHz so that an antenna with adequate gain can be mounted on the aircraft (9).
- The radar is locked on to the target so that multiple range profiles will be available for target identification in a relatively short period of time (16).

1.6 Approach and Methodology

Software systems are currently available which – when given a detailed description of a complex target such as an aircraft – can accurately predict the range profile of the target at an arbitrary aspect angle (3). However, the reverse process by which the detailed structure of a complex target is determined at an arbitrary aspect angle from its range profile cannot be reliably performed. The reverse process is difficult to perform because scatterers at a given range have an arbitrary phase so that they reinforce or cancel stochastically. Also, scatterers may be illuminated by more than one signal path as energy reflects from one scatterer to another before returning to the radar. Both of the processes described above are very dependent on aspect angle. In general, processes which are too complex to be represented by a deterministic model must be modeled as *random phenomenon* (6). Therefore, the RTI system developed in this thesis will be based on the stochastic properties of the range profiles and how the range profiles change with aspect angle.

1.6.1 Range Profile Feature Extraction and Vector Quantization Range profiles will be processed using the Prony Technique developed by Steedly and Sacchini (25) (27). Using this technique each scattering center is described by a polarization ellipse, relative energy, frequency response and range. The data is reduced by vector quantizing to clustering centers. The clustering centers will be chosen using a K-means clustering algorithm.

1.6.2 Target Identification In general, target identification will made based on the temporal relationships between scattering centers of single range profiles as well as the temporal relationships between sequences of range profiles. One HMM per class will be used to model single range profiles over the entire aspect angle window of interest. Identification of unknown targets will be realized by determining the HMM which has the highest probability of generating the unknown target's range profile. If identification is made based on sequences of range profiles, multiple HMM's per class will

be used and the identification decision will be reached according to which group of HMM's has the highest probability of generating the sequence of range profiles.

1.6.3 HRR Radar Signature Synthesis Actual HRR radar signatures are not readily available. Therefore, synthesized HRR radar returns will be used for this thesis. A number of software packages that generate HRR radar range profiles are available. One such package is Xpatch which calculates the frequency-domain RCS of a faceted target by geometric optical ray casting with multiple bounce technique (3). Range profiles produced by Xpatch will form the data set for this research.

The Xpatch range profiles are fully polarized, in that four range profiles are generated for each signature calculation: vertical transmit/vertical receive; vertical transmit/horizontal receive; horizontal transmit/horizontal receive; and horizontal transmit/horizontal receive. The four range profiles are represented in the frequency domain and consist of 1024 discrete frequency samples from approximately 9.25 to 10.75 GHz. In the time-domain, the range profiles are divided into 0.1 meter range bins covering a total of 102.4 meters per range profile. The incident plane wave used to calculate the signatures in Xpatch is non-causal and therefore, is not physically realizable in hardware(18).

In other words, the signature generated by Xpatch is essentially the impulse response of the target over the 1.5 GHz bandwidth. Therefore, more realistic signatures can be obtained by adding white Gaussian noise to the signatures and convolving them with a transfer function of a causal HRR radar such as a Linear Frequency Modulated pulse compression (26) HRR radar whose bandwidth is less then 1.5 GHz.

1.7 Research Objectives

This thesis will investigate means to exploit HRR radar range profiles for the purpose of target identification. The focus of the research will be on an efficient and reliable means to represent a given target using Prony's Model and HMMs. The specific objectives of this research effort are follows:

- To develop a reliable RTI algorithm which is designed to identify targets using a Linear FM pulse compression HRR radar as a sensor.
- To characterize the RTI algorithm by measuring its susceptibility to additive white Gaussian noise (AWGN) and range alignment errors caused by imperfections in the tracking system.

1.8 Thesis Organization

The organization of this thesis is given below.

- The detailed background material pertaining to the major components of the RTI algorithm developed for this thesis is reviewed in Chapter II.
- The RTI algorithm description is found in Chapter III.
- Chapter IV contains the experimental procedures and results.
- The conclusion and recommendations are given in Chapter V.

1.9 Summary

In the fast paced environment of air-to-air combat, positive and timely target identification is critical. Although HRR radar is possible with current technology, RTI using HRR radar signatures is an unsolved problem. Current research has shown that features based on the polarization of the target's scattering centers correspond well to the physical structures of the target. Therefore, these features should be useful for RTI. Given an accurate feature set and a reliable identification algorithm, RTI in a realistic combat environment will become a reality.

II. BACKGROUND

2.1 Introduction

Radar Target Identification (RTI) research spans a broad range of radar types and classification algorithms. This background material presented in this chapter will focus on RTI based on High Range Resolution (HRR) radar signatures. The following areas will be reviewed:

- HRR radar
- High resolution exponential modeling of fully polarized radar returns
- Hidden Markov Models
- Classification performance clarification

2.2 High Range Resolution (HRR) Radar

As stated in Chapter I, HRR radar is specifically designed to resolve the target into many parts as a function of range along the radar-target vector. The range from the radar to the target along the radar-target vector is determined by measuring the time delay between the transmitted signal and the received backscattered signal from the target. The range is computed using the propagation velocity of the transmitted signal. Hence, the range is given by

$$r = \frac{ct_d}{2} \tag{2.1}$$

$r \equiv$ range: m

$c \equiv$ speed of light in free space: 3×10^8 m/s

$t_d \equiv$ time delay: s.

The 2 in the denominator accounts for the propagation of the transmitted signal to and from the target. If a certain radar transmits a rectangular pulse with a pulse width τ seconds, the range increment or range resolution is $\delta r = c\tau/2$, and two point reflectors separated by distances greater than δr can be resolved as two distinct targets. The bandwidth B of a τ second rectangular pulse is approximately

$B = 1/\tau$, and the range resolution is usually expressed in terms of B so that (26)

$$\delta r = \frac{c}{2B} \tag{2.2}$$

$\delta r \equiv$ range resolution

$B \equiv$ bandwidth of the transmitted signal.

Obviously, HRR can be achieved by transmitting a very narrow pulse ($\tau = 667$ ps for $\delta r = 0.1$ m), but the maximum range of a narrow pulse HRR radar is limited by the low energy of the transmitted pulse.

Therefore, HRR radar design is constrained by two opposing parameters: the energy and bandwidth of the transmitted pulse. Increasing the transmitted pulse width increases the energy but also decreases the bandwidth which in-turn decreases the range resolution. Two schemes for realizing HRR radar capable of detecting targets at ranges suitable for RTI are discussed below.

2.2.1 Synthetic HRR Radar As shown by Equation (2.2) , δr is limited by B. Synthetic HRR radar realizes the necessary bandwidth for HRR by measuring the steady state response of the target at discrete frequencies over the required bandwidth. Thus, the frequency domain response of the target is represented by a set of scattering coefficients which are measured at discrete frequencies

$$S\left[(f_k - f_{\min})\Delta f\right] = S[k],\ k = 0, 1, 2, ..., N-1 \tag{2.3}$$

$k \equiv$ integer index

$N \equiv$ number of samples

$f_k \equiv$ frequency

$f_{\min} \equiv$ minimum frequency of the sequence

$f_s \equiv$ sampling rate in the discrete time domain (at least the bandpass Nyquist rate)

$\Delta f \equiv$ frequency step size: $\frac{f_s}{N}$

the range resolution expressed in terms of Δf and N is

$$\delta r = \frac{c}{2(N-1)\Delta f} \tag{2.4}$$

The range profile of the target is the inverse Discrete Fourier Transform (DFT) of $S[k]$, or

$$s[n] = \frac{1}{N}\sum_{k=0}^{N-1} S[k]e^{\frac{j2\pi kn}{N-1}}, \quad n = 0, 1, 2, ..., N-1. \tag{2.5}$$

In the discrete time domain, $t_d = n/f_s$ and the discrete range is

$$r_n = \frac{cn}{2f_s}. \tag{2.6}$$

While the classical definition of unambiguous range is related to the pulse repetition time (prt) or the time between transmitted pulses of the radar, because $s[n]$ is periodic with a period of N, the maximum unambiguous range for synthetic HRR radar is limited to

$$R_E = \frac{cN}{2f_s} = \frac{c}{2\Delta f}. \tag{2.7}$$

If a range resolution of 0.1 meters is desired, by Equation (2.2) B must be 1.5 GHz and f_s– for the bandpass Nyquist rate – must also be at least 1.5 GHz. For $N = 1024$, the maximum unambiguous range R_E is only 102.4 meters. If, for example, the target is at 50 Km, $S[k]$ must be measured at the proper time delay. That is, $S[k]$ is windowed in time by a window that is centered on the target. However, the range extent of the target must be less then R_E to prevent aliasing in the discrete range domain.

Synthetic HRR radar is very useful for test facilities where the target is stationary because it essentially measures the finite bandwidth impulse response of the target. However, the utility of synthetic HRR radar for RTI of aircraft outside the laboratory is limited because the aircraft would generally be at a different location and aspect angle for each frequency measurement. It is possible, however, to approximate the impulse response of the target over a finite bandwidth if the transmitted signal is relatively wide pulse which is continuously swept across the frequency band (26). This process is described in the following section.

2.2.2 Linear FM Pulse Compression As stated previously, HRR design is constrained by the energy and bandwidth of the transmitted pulse. Pulse compression HRR radars realize HRR by modulating a relatively wide, and therefore, high energy pulse with a signal that has symmetric, narrow peaked autocorrelation properties. Two such examples of signals with these autocorrelation properties are the Barker code and linear FM modulation waveforms (26). What follows is a description of pulse compression using a linear FM modulated or 'chirped' transmitted pulse employed by many modern HRR radars (26). The derivations of the linear FM pulse compression equations stated below can be found in reference (5). Analytically, the transmitted pulse is

$$\begin{aligned} f(t) &= \cos\left(2\pi f_c\left[t-\frac{T}{2}\right]+\int\frac{2\pi B_{fm}}{T}\left[t-\frac{T}{2}\right]dt\right)\text{rect}\left(\frac{\left[t-\frac{T}{2}\right]}{T}\right) \\ &= \cos\left(\phi(t)\right)\text{rect}\left(\frac{\left[t-\frac{T}{2}\right]}{T}\right) \end{aligned} \tag{2.8}$$

$f_c \equiv$ carrier frequency

$T \equiv$ pulse width

$B_{fm} \equiv$ bandwidth

The instantaneous frequency for $0 \leq t \leq T$ is $\frac{d\phi(t)}{dt}$ or

$$f_{inst}(t) = f_c + \frac{B_{fm}}{T}\left(t-\frac{T}{2}\right) \quad 0 \leq t \leq T. \tag{2.9}$$

Therefore, the frequency of the pulse is linearly swept from $(f_c - B_{fm}/2)$ to $(f_c + B_{fm}/2)$. Taking the Fourier Transform of $f(t)$ to represent the transmitted signal in the frequency-domain gives

$$\begin{aligned} F(f) &= \frac{1}{2}\sqrt{\frac{\pi}{U}}\left\{[C(x_1)+C(x_2)]^2+[S(x_1)+S(x_2)]^2\right\}^{\frac{1}{2}} \\ &\quad \exp\left\{-\frac{j}{2U}(2\pi f_c-2\pi f)^2+j\tan^{-1}\left(\frac{S(x_1)+S(x_2)}{C(x_1)+C(x_2)}\right)\right\} \end{aligned} \tag{2.10}$$

where

$$
\begin{aligned}
U &\equiv \frac{2\pi B_{fm}}{T} \\
x_1 &\equiv \frac{\frac{T}{2} + 2\pi(f_c - f)}{\sqrt{\pi U}} \\
x_2 &\equiv \frac{\frac{T}{2} - 2\pi(f_c - f)}{\sqrt{\pi U}}
\end{aligned}
$$

and $S(x)$ and $C(x)$ are Fresnel integrals

$$
\begin{aligned}
C(x) &\equiv \int_0^x cos\left(\frac{\pi}{2}\alpha^2\right) d\alpha \\
S(x) &\equiv \int_0^x sin\left(\frac{\pi}{2}\alpha^2\right) d\alpha.
\end{aligned}
$$

Recall that the transmitted signal, $F(f)$, has symmetric, narrow peaked autocorrelation properties. Thus, filtering the received signal with a filter matched to $F(f)$ compresses the the received pulses. The matched filter output, which is denoted as $S(f) = F(f)F^*(f)$, is a sampling function in the discrete time domain whose first null is at $1/B_{fm}$. Therefore, the range resolution is $c/2B_{fm}$ and the transmitted pulse appears to be an unmodulated rectangular pulse which has a pulse width of $\tau = 1/B_{fm}$ seconds. Hence, the pulse compression ratio is expressed as τ/T. In general, a higher pulse compression ratio decreases the magnitude of U which increases the sharpness of the roll off of $F(f)$ (5).

2.3 High Resolution Exponential Modeling of Fully Polarized Radar Returns

Scattering centers along the range profile of HRR radar signatures can be modeled parametrically by an exponential model (25) (27). This model is referred to as Prony's model. Using Prony's model, the range profile of the target is described by T scattering centers, each characterized by $|p_{t_n}|$ (frequency response), AR_{t_n} (polarization ellipse axis ratio), τ_{t_n} (polarization ellipse tilt angle), E_{t_n} (energy), and r_{t_n} (range). The parameter estimation techniques developed by Sacchini and Steedly have overcome the adverse affects of noise which usually limits the usefulness of the Prony model (25) (27).

It is assumed that the the frequency domain response of the target is represented by a fully polarized set of scattering coefficients which are measured (for synthetic HRR) or sampled (for Linear

FM Pulse Compression HRR) at discrete frequencies $k = 0, 1, 2, ..., N-1$. These coefficients are denoted as $S_{vv}[k]$, $S_{vh}[k]$, $S_{hv}[k]$, and $S_{hh}[k]$, where the first subscript corresponds to the receive polarization (h for horizontal and v for vertical polarization) and the second subscript corresponds to the transmit polarization. The four transmit/receive polarization combinations can be used to synthesize any arbitrary polarization desired. Left circular polarization is synthesized by the following transform (29):

$$\begin{pmatrix} S_{hl}[k] \\ S_{vl}[k] \end{pmatrix} = \begin{pmatrix} S_{hh}[k] & S_{hv}[k] \\ S_{vh}[k] & S_{vv}[k] \end{pmatrix} \begin{pmatrix} 1 \\ j \end{pmatrix} \frac{1}{\sqrt{2}}. \tag{2.11}$$

The subscripts (hl) and (vl) on the left side of Equation (2.11) correspond to a horizontally received and vertically received left circularly polarized transmitted signal.

As shown in Section 2.2.1, the range profile, $s_{xy}[r_n]$ (xy is hl or vl), can be found using Equations (2.5) and (2.6), and Equation (2.5) is efficiently computed with a Fast Fourier Transform (FFT) routine. Except for computational cost, the performance of the Prony model is superior to the conventional Inverse Fast Fourier Transform (IFFT) techniques because it yields higher range resolution for the same bandwidth and it models the frequency response of individual scattering centers. Using the IFFT to calculate range profiles, each range resolution cell (n index) is modeled with a constant frequency response over all frequencies $k = 0, 1, 2, ..., N-1$ because the kernel of (2.5) has a magnitude of 1. Although a perfect sphere has a constant frequency response, realistic scattering centers do not display a constant frequency response. For example, the frequency response of a trihedral increases linearly as frequency increases (25). In this section the definitions of the Prony model parameters and the procedure to estimate these parameters is presented.

2.3.1 Prony's Model The Prony model (23) of the fully polarized frequency data given in Equation (2.11) is

$$\begin{pmatrix} S_{hl}[k] \\ S_{vl}[k] \end{pmatrix} = \sum_{n=1}^{T} \begin{pmatrix} a_{ht_n} \\ a_{vt_n} \end{pmatrix} p_{t_n}^k, \quad k = 0, 1, 2, ..., N-1. \tag{2.12}$$

where

p_{t_n} $\equiv$ pole of the n^{th} scattering center (generally complex)

a_{ht_n} $\equiv$ amplitude coefficient of the horizontally received n^{th} scattering center

a_{vt_n} $\equiv$ amplitude coefficient of the vertically received n^{th} scattering center

T $\equiv$ number of scattering centers

N $\equiv$ number of discrete frequencies

Clearly, Equation (2.12) is similar to Equation (2.5). If the n^{th} pole corresponds to an ideal point scatterer, $|p_{t_n}| = 1$, but for more realistic scatterers with frequency responses that are not constant, $|p_{t_n}|$ will vary slightly around 1. The relative range of the n^{th} scattering center is associated with the angle of p_{t_n}, $\angle p_{t_n}$, and is computed as

$$r_{t_n} = R_E \frac{\angle p_{t_n}}{2\pi}, \quad 0 \leq r_{t_n} \leq R_E. \tag{2.13}$$

R_E is the unambiguous range given by Equation (2.7). The energy of the n^{th} scatterer is calculated as

$$E_{t_n} = (|a_{ht_n}|^2 + |a_{vt_n}|^2) \sum_{k=0}^{N-1} |p_{t_n}|^{2k} \quad n = 1, 2, \cdots, T \tag{2.14}$$

The polarization information is contained in the vertical and horizontal amplitude coefficients, a_{ht_n} and a_{vt_n}. The polarization of the n^{th} scattering center is an ellipse which can be represented by the Poincaré Sphere (14). The polarization ellipse parameters are calculated with the following equations:

$$\gamma_{t_n} = \tan^{-1}\left(\frac{a_{vt_n}}{a_{ht_n}}\right), \quad 0 \leq \gamma_{t_n} \leq \frac{\pi}{2} \tag{2.15}$$

$$\delta_{t_n} = \angle a_{vt_n} - \angle a_{ht_n}, \quad -\pi \leq \delta_{t_n} \leq \pi \tag{2.16}$$

$$\epsilon_{t_n} = \frac{1}{2}\sin^{-1}\left[\sin(2\gamma_{t_n})\sin(\delta_{t_n})\right], \quad -\frac{\pi}{4} \leq \epsilon_{t_n} \leq \frac{\pi}{4} \tag{2.17}$$

A complete and compact description of the shape of the polarization ellipse is given by the tilt angle, τ_{t_n}, and the major to minor axis ratio AR_{t_n}.

$$\tau_{t_n} = \frac{1}{2}\tan^{-1}\left[\tan(2\gamma_{t_n})\cos(\delta_{t_n})\right], \quad 0 \leq \tau_{t_n} \leq \frac{\pi}{2} \tag{2.18}$$

$$AR_{t_n} = \cot(\epsilon_{t_n}), \quad -\infty \leq AR_{t_n} \leq +\infty \tag{2.19}$$

Although τ_{t_n} ranges from 0 to π, τ_{t_n} is limited to $\pi/2$ in (2.18) because only one quarter of the Poincaré Sphere is used. To avoid this ambiguity, the following alterations to τ_{t_n} need to be made (27):

$$\tau_{t_n} = \begin{cases} \tau_{t_n} + \frac{\pi}{2} & \text{if} \quad \gamma_{t_n} \geq \frac{\pi}{4} \\ \tau_{t_n} + \pi & \text{if} \quad \gamma_{t_n} \leq \frac{\pi}{4} \quad \text{and} \quad \tau_{t_N} < 0 \end{cases} \tag{2.20}$$

For left-handed polarization $\epsilon_{t_n} \geq 0$ and $0 \leq AR_{t_n} \leq +\infty$.

The set of parameters $\{|p_{t_n}|, AR_{t_n}, \tau_{t_n}, E_{t_n}, \tau_{t_n}; n = 1, 2 \cdots, T\}$ forms a concise description of each scattering center of the target. The target is described by T such scattering centers.

2.3.2 Parameter Estimation Parameter estimation is required to estimate the poles and amplitude coefficients from the frequency data. The following algorithm for estimating the poles and amplitude coefficients was developed at Ohio State University (25) (27). The poles are estimated using a backward linear prediction approach which uses a Singular Value Decomposition (SVD). The backward linear prediction equation simultaneously uses both the horizontal and vertical polarizations

and is written as

$$\begin{pmatrix} S_{hl}[0] & S_{hl}[1] & S_{hl}[2] & \dots & S_{hl}[Q] \\ S_{hl}[1] & S_{hl}[2] & S_{hl}[3] & \dots & S_{hl}[Q+1] \\ \vdots & \vdots & \vdots & & \vdots \\ S_{hl}[N-1-Q] & S_{hl}[N-Q] & S_{hl}[N-Q+1] & \dots & S_{hl}[N-1] \\ S_{vl}[0] & S_{vl}[1] & S_{vl}[2] & \dots & S_{vl}[Q] \\ S_{vl}[1] & S_{vl}[2] & S_{vl}[3] & \dots & S_{vl}[Q+1] \\ \vdots & \vdots & \vdots & & \vdots \\ S_{vl}[N-1-Q] & S_{vl}[N-Q] & S_{vl}[N-Q+1] & \dots & S_{vl}[N-1] \end{pmatrix} \begin{pmatrix} 1 \\ b_1 \\ b_2 \\ \vdots \\ b_Q \end{pmatrix} \approx 0 \quad (2.21)$$

or

$$\boldsymbol{S} \begin{pmatrix} 1 \\ \boldsymbol{b} \end{pmatrix} \approx 0. \quad (2.22)$$

K is the number of discrete frequencies, Q is the prediction order, and $\boldsymbol{b}$ is the coefficient vector of the polynomial $B(z)$ given by

$$B(z) = 1 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_Q z^{-Q} \quad (2.23)$$

$1/B(z)$ describes an all pole filter in the z-transform domain that has Q poles. The prediction order, Q, is ideally an integer greater than the model order T. Prior to solving $\boldsymbol{b}$, $\boldsymbol{S}$ is expanded by a SVD so that

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^H \quad (2.24)$$

where $\boldsymbol{V}^H$ denotes the Hermitian transpose (or conjugate transpose) of $\boldsymbol{V}$ and

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1 & & & 0 \\ & \sigma_2 & & \\ & & \ddots & \\ 0 & & & \sigma_{Q+1} \end{pmatrix}. \quad (2.25)$$

The singular values of S are ordered form largest to smallest ($\sigma_1 > \sigma_2 > \ldots > \sigma_{Q+1}$). S is noise cleaned by keeping the first T singular values and setting the remaining singular values to zero so that

$$\hat{\Sigma} = \begin{pmatrix} \sigma_1 & & & & & & 0 \\ & \sigma_2 & & & & & \\ & & \ddots & & & & \\ & & & \sigma_T & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ 0 & & & & & & 0 \end{pmatrix}. \tag{2.26}$$

Note that $\hat{\Sigma}$ has the same dimensions as Σ. Now the noise cleaned version, $\hat{\mathrm{S}}$, is formed by

$$\hat{S} = U\hat{\Sigma}V^H \tag{2.27}$$

Now $\widehat{S}$ is written as $[\widehat{S}_1\ \widehat{S}_2]$ where $\widehat{S}_1$ is a column vector consisting of the first column of $\widehat{S}$ and $\widehat{S}_2$ is a matrix consisting of the remaining columns of $\widehat{S}$. The estimate of b, $\hat{b}$, is then found by using a least squares solution, which is written as

$$\hat{b} = -\left(\widehat{S}_2^H \widehat{S}_2\right)^{-1} \widehat{S}_2^H \widehat{S}_1. \tag{2.28}$$

Finally, the estimated poles are found by solving for the roots of $\hat{B}(z)$ as

$$\hat{p}_q = \frac{1}{\mathrm{root}_q(\hat{B}(z))}, \quad q = 1, 2, \ldots, Q. \tag{2.29}$$

Although there are Q roots of $\hat{B}(z)$, only T of those roots corresponds to data modes because only T singular values of $\hat{\mathrm{S}}$ are nonzero. Determining which of the Q poles are valid is accomplished in two steps. First, poles that do not fit the following criteria are discarded.

$$\frac{1}{100} < |\hat{p}_q|^N < 100. \tag{2.30}$$

The above criteria has been found to model radar data well (27). After the invalid poles have been discarded, the T poles with highest energy are retained. From Equation (2.14), the energy calculation requires both the horizontal and vertical amplitude coefficients.

Writing Equation (2.12) in matrix form gives

$$\begin{pmatrix} \hat{p}_1^1 & \hat{p}_2^1 & \cdots & \hat{p}_{Q'}^1 \\ \hat{p}_1^2 & \hat{p}_2^2 & \cdots & \hat{p}_{Q'}^2 \\ \vdots & \vdots & \vdots & \vdots \\ \hat{p}_1^N & \hat{p}_2^N & \cdots & \hat{p}_{Q'}^N \end{pmatrix} \begin{pmatrix} \hat{a}_{h1} & \hat{a}_{v1} \\ \vdots & \vdots \\ \hat{a}_{hQ'} & \hat{a}_{vQ'} \end{pmatrix} = \begin{pmatrix} S_{hl}[0] & S_{vl}[0] \\ \vdots & \vdots \\ S_{hl}[N-1] & S_{vl}[N-1] \end{pmatrix} \tag{2.31}$$

or

$$\hat{P}_{Q'} \hat{A}_{Q'} = S_a \tag{2.32}$$

Q' is the number of poles which satisfy the criterion of Equation (2.30). $\hat{A}$ is found using a least square solution as

$$\hat{A}_{Q'} = \left(\hat{P}_{Q'}^H \hat{P}_{Q'} \right)^{-1} \hat{P}_{Q'}^H S_a. \tag{2.33}$$

The energy of each of the Q' poles is calculated using Equation (2.14) and the T poles with the highest energy are retained. The parameter estimation procedure is terminated by reestimating the amplitude coefficients of the T true poles such that

$$\hat{A}_T = \left(\hat{P}_T^H \hat{P}_T \right)^{-1} \hat{P}_T^H S_a. \tag{2.34}$$

2.4 Hidden Markov Models

HMM's have been explored by a number of researchers since 1975 (19). Rabiner (11) (13) (20) (21)(22) and Levinson (15) have written HMM tutorials which have been summarized in *Voice and Speech Processing* by Parsons (19). The purpose of this section is to present the general HMM theory applicable to RTI. Most of the HMM theory in this section is drawn from Rabiner's, *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition* (22) and the notation will, for the most part, correspond to Rabiner's notation in that article.

2.4.1 HMM Background Real-world processes produce observable outputs which are characterized as signals. In a broad sense, signal models can be separated into two classes: deterministic and statistical. A sine wave, for example, is characterized by a deterministic signal model that is completely specified by amplitude, frequency, and phase; all of which are fixed parameters. On the other hand, statistical signal models characterize the signal's statistical properties. Gaussian and Poisson signal models are two well known statistical signal models (6). The underlying assumption of any statistical model is that the signal can be well characterized as a parametric random process, and that the parameters of this stochastic process can be determined (estimated) in a precise, well-defined manner (22).

The HMM is also a statistical signal model and is defined as (21),

> "... a doubly stochastic process with an underlying stochastic process that is not observed (it's hidden), but can only be observed through another set of stochastic processes that produce the sequence of observed symbols."

In this section the theory of Markov chains will be presented and then extended to HMM's. After the concepts of HMM's are introduced three fundamental problems for HMM design will be discussed in detail.

2.4.2 Markov Chains A Markov chain is a system which is described at any time as being in one of a set of N distinct states, $S_1, S_2, \ldots, S_N$. The system experiences a change of state (possibly back to the same state) at evenly spaced discrete times $t = t_1, t_2, \ldots, t_n$. The active state at time t_n is denoted as q_{t_n}. The next state, $q_{t_{n+1}}$, is determined by the state transition probabilities associated with the current state, q_{t_n}. The state transition probabilities, a_{ij}, are written as

$$a_{ij} = P\left(q_{t_{n+1}} = S_j | q_{t_n} = S_i\right), \quad 1 \leq i, j \leq N. \tag{2.35}$$

Where N is the number of states, S_i is the current state, and S_j is the next state. The state transition coefficients must satisfy standard stochastic constraints so that

$$0 \leq a_{ij} \leq 1 \tag{2.36}$$

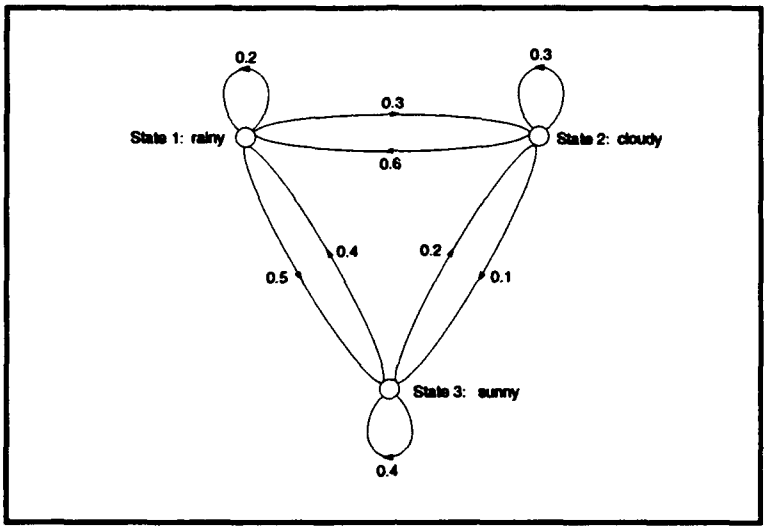


Figure 2.1. Three State Markov Chain to Model the Weather

$$\sum_{j=1}^{N} a_{ij} = 1. \tag{2.37}$$

Each state in the above process corresponds to an observable, physical event. Rabiner clarifies the concepts of the Markov chain by illustrating a simple 3-state model of the weather which is depicted in Figure 2.1 (22) . The weather is observed at the same time each day as one of the following:

State 1: rainy or (snowy)

State 2: cloudy

State 3: sunny.

The state transition probabilities for times $t_n \geq t_1$ are compactly represented by the A matrix as follows

$$A = (a_{ij}) = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}. \tag{2.38}$$

The initial state probabilities at time t_1 are similarly represented with the following notation

$$\pi_i = P(q_{t_1} = S_i), \quad 1 \leq i \leq N \tag{2.39}$$

and are also represented in matrix format using the column matrix π as follows:

$$\pi = (\pi_i) = \begin{pmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \end{pmatrix} \tag{2.40}$$

Now consider the following weather observations over a 5 day period, "rainy – sunny – cloudy – sunny – rainy". More formally the observation sequence O is stated as $O = \{S_1, S_3, S_2, S_3, S_1\}$. The state transitions are independent by the Markov property (22), so the probability of O given the model is expressed as

$$P(O|\text{Model}) = P(S_1, S_3, S_2, S_3, S_1|\text{Model}) \tag{2.41}$$

$$= P(S_1) \cdot P(S_3|S_1) \cdot P(S_2|S_3) \cdot P(S_3|S_2) \cdot P(S_1|S_3) \tag{2.42}$$

$$= \pi_1 \cdot a_{13} \cdot a_{32} \cdot a_{23} \cdot a_{31} \tag{2.43}$$

Before extending these ideas to HMM's, it is important to understand that the states of the Markov chain are explicitly specified by the observed weather. That is to say that if the weather is observed as *sunny*, the model is in state 3 (S_3). Therefore, the Markov chain is completely visible from

the observed sequence. As will be described in the next section, the states of a HMM are *not* specified by the observed sequence – they are hidden.

2.4.3 Extension to HMM's Rabiner introduces the concepts of HMM's using several biased coin toss experiments and the classic urn and ball experiment (21) (22). Instead of summarizing those experiments here, an experiment using weighted die will be demonstrated to provide a broader understanding of basic HMM concepts, when combined with Rabiner's examples.

Consider a system which produces an integer number between 1 and 6 at evenly spaced time increments. A typical observed sequence might be

$$\begin{aligned} O &= O_{t_1} O_{t_2} O_{t_3} \cdots O_{t_T} \\ &= 3\,4\,1 \cdots 2. \end{aligned}$$

Given, the above set of observations, the problem of interest is to build a HMM to explain the observed sequence of integers (22). Obviously, there are numerous mechanisms which could account for the above observations. One of which is a single, fair die, but for this example, let the system be comprised of 3 biased die which are hidden from the observer by a barrier. Each die is biased as follows:

Die 1		**Die 2**		**Die 3**	
$P_1(1)$	$= 0.3$	$P_2(1)$	$= 0.05$	$P_3(1)$	$= 0.7$
$P_1(2)$	$= 0.15$	$P_2(2)$	$= 0.1$	$P_3(2)$	$= 0.025$
$P_1(3)$	$= 0.15$	$P_2(3)$	$= 0.6$	$P_3(3)$	$= 0.1$
$P_1(4)$	$= 0.1$	$P_2(4)$	$= 0.05$	$P_3(4)$	$= 0.05$
$P_1(5)$	$= 0.2$	$P_2(5)$	$= 0.1$	$P_3(5)$	$= 0.025$
$P_1(6)$	$= 0.1$	$P_2(6)$	$= 0.1$	$P_3(6)$	$= 0.1$

The above weights were chosen arbitrarily, but the combination of weights for each die must sum to 1.

According to some random process, an initial die is chosen and rolled. The outcome of the roll is recorded as the first observation at time t_1. A new die is then chosen (possibly the same die) according to the random process selection associated with the current die. At the next time increment, t_2, the new

die is rolled and its output is recorded as the second observation. A new dice is chosen based on the current die and the process is repeated.

In a straight forward way the system can be modeled with a three state ($N = 3$) HMM, one for each die. The initial state (the die chosen at t_1) and the state transitions (the new die chosen based on the current die) can be represented as the π (2.40) and A (2.38) matrices introduced for Markov chains in the previous section. For this example let

$$\pi = (\pi_i) = \begin{pmatrix} 0.4 \\ 0.3 \\ 0.3 \end{pmatrix} \tag{2.44}$$

and

$$A = (a_{ij}) = \begin{pmatrix} 0.3 & 0.5 & 0.2 \\ 0.6 & 0.2 & 0.2 \\ 0.1 & 0.2 & 0.7 \end{pmatrix}. \tag{2.45}$$

Clearly, the structure of the HMM is similar to that of the Markov chain. The primary difference between the two is that the observation outputs do not uniquely correspond to any given state. Therefore, to fully describe a HMM the observation probabilities must be specified for each state. The following notation is used

$$b_i(m) = P(v_m \text{ at } t_n | q_{t_n} = S_i), \quad 1 \leq i \leq N \quad 1 \leq m \leq M \tag{2.46}$$

Where v_m is the m^{th} element of the output symbol set, $V = \{v_1, v_2, \cdots, v_M\}$, and M is the number of symbols in the output symbol set. For the current example, $M = 6$, $v_1 = 1$, $v_2 = 2$, $\cdots$, and $v_6 = 6$. As with A and π the $b_i(m)$'s are also written as elements of a B matrix. The probabilistic weights of the dice are expressed in the B matrix as

$$B = (b_i(m)) = \begin{pmatrix} 0.3 & 0.15 & 0.15 & 0.1 & 0.2 & 0.1 \\ 0.05 & 0.1 & 0.6 & 0.05 & 0.1 & 0.1 \\ 0.7 & 0.025 & 0.1 & 0.05 & 0.025 & 0.1 \end{pmatrix} \tag{2.47}$$

The HMM is completely specified by N, M, $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{\pi}$ and for convenience the compact notation

$$\boldsymbol{\lambda} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi}) \tag{2.48}$$

is used.

The model, $\boldsymbol{\lambda}$, is generally used to describe an observed sequence but, it can also be used to generate an observation sequence $\boldsymbol{O}$ that has T observations as follows (22):

1. Choose an initial state $q_{t_1} = S_i$ according to the initial state distribution $\boldsymbol{\pi}$.
2. Set $t = t_1$.
3. Choose $O_{t_1} = v_m$ according to the symbol probability distribution in state S_i.
4. Transition to a new state state $q_{t_{n+1}} = S_j$ according to the state transition probability distribution for state S_i.
5. Set $t = t_{n+1}$; return to step (3) if $t \leq t_T$; otherwise, terminate the procedure.

This procedure was used to generate 3 observation sequences, $\boldsymbol{O}^k$, with $T = 10$ observations each. The superscript k denotes the sequence number. The 3 observation sequences are

$$\begin{aligned}
\boldsymbol{O}^1 &= 5\ 3\ 1\ 3\ 4\ 3\ 3\ 5\ 5\ 1 \\
\boldsymbol{O}^2 &= 2\ 3\ 2\ 3\ 3\ 4\ 6\ 1\ 5\ 6 \\
\boldsymbol{O}^3 &= 3\ 3\ 6\ 3\ 2\ 5\ 1\ 1\ 1\ 1.
\end{aligned}$$

From the perspective of the observer, who can only see the observation sequences, the state of the model at $t = t_n$ as well as the weights of the individual dice are not readily apparent.

2.4.4 HMM Design As previously mentioned there are three fundamental problems for HMM design. These problem are given below using the notation presented in the previous sections (21) (22).

1. Given the observation sequence $\boldsymbol{O} = O_{t_1}\ O_{t_2}\ \cdots\ O_{t_T}$, and a model $\boldsymbol{\lambda} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi})$. how can $P(\boldsymbol{O}|\boldsymbol{\lambda})$, the probability of the observation sequence, given the model, be calculated efficiently?

2. Given the observation sequence $\boldsymbol{O} = O_{t_1}\ O_{t_2}\ \cdots\ O_{t_T}$, and a model $\boldsymbol{\lambda}$, how can a corresponding state sequence $\boldsymbol{Q} = q_{t_1}\ q_{t_2}\ \cdots\ q_{t_T}$ be chosen, which is optimal in some meaningful sense (i.e., best "explains" the observations)?

3. How can the model parameters, $\boldsymbol{\lambda} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi})$, be chosen to maximize $P(\boldsymbol{O}|\boldsymbol{\lambda})$?

2.4.5 Observation Probability Given the Model The probability of an observation sequence $\boldsymbol{O}$, given the model $\boldsymbol{\lambda}$ ($P(\boldsymbol{O}|\boldsymbol{\lambda})$) is a measure of how well a given model matches an observation sequence. Therefore, $P(\boldsymbol{O}|\boldsymbol{\lambda})$, is a way of choosing the best model among several competing models (21) (22). For isolated word recognition (IWR), a HMM is synthesized for each word to be identified. The identification of an utterance of an unknown word is accomplished by choosing the word whose HMM has the highest $P(\boldsymbol{O}|\boldsymbol{\lambda})$.

$P(\boldsymbol{O}|\boldsymbol{\lambda})$ is computed in a straight forward way by taking the sum of the joint probabilities of every state sequence $\boldsymbol{Q}^l = q_{t_1}\ q_{t_2}\ \cdots\ q_{t_T}$ (21) and the observation sequence $\boldsymbol{O}$ of interest. $P(\boldsymbol{O}, \boldsymbol{Q}^l|\boldsymbol{\lambda})$ can be computed as the product of $P(\boldsymbol{O}|\boldsymbol{Q}^l, \boldsymbol{\lambda})$ and $P(\boldsymbol{Q}^l|\boldsymbol{\lambda})$. It follows that

$$P(\boldsymbol{O}, \boldsymbol{Q}^l|\boldsymbol{\lambda}) = P(\boldsymbol{O}|\boldsymbol{Q}^l, \boldsymbol{\lambda}) \cdot P(\boldsymbol{Q}^l|\boldsymbol{\lambda}). \tag{2.49}$$

The second term on the right side of (2.49) is calculated exactly the same way as $P(\boldsymbol{O}|\mathbf{Model})$ in Equation (2.41) for the Markov chain such that

$$P(\boldsymbol{Q}^l|\boldsymbol{\lambda}) = \pi_{q_{t_1}} a_{q_{t_1} q_{t_2}} a_{q_{t_2} q_{t_3}} \cdots a_{q_{t_{T-1}} q_{t_T}}. \tag{2.50}$$

The calculation of $P(\boldsymbol{O}|\boldsymbol{Q}^l, \boldsymbol{\lambda})$ is also straight forward.

$$P(\boldsymbol{O}|\boldsymbol{Q}^l, \boldsymbol{\lambda}) = b_{q_{t_1}}(O_{t_1}) b_{q_{t_2}}(O_{t_2}) \cdots b_{q_{t_T}}(O_{t_T}). \tag{2.51}$$

Finally, $P(\boldsymbol{O}|\boldsymbol{\lambda})$ is obtained by summing $P(\boldsymbol{O}, \boldsymbol{Q}^l|\boldsymbol{\lambda})$ over all possible state sequences:

$$P(\boldsymbol{O}|\boldsymbol{\lambda}) = \sum_{\text{all } l} P(\boldsymbol{O}|\boldsymbol{Q}^l, \boldsymbol{\lambda}) \cdot P(\boldsymbol{Q}^l|\boldsymbol{\lambda}). \tag{2.52}$$

Although the calculation of $P(O|\lambda)$ directly using (2.52) is straight forward, it is computationally unfeasible even for small values of N and T, since there are $(2T-1)$ multiplications per state sequence, N^T possible state sequences, and $N^T - 1$ additions (21). For $N = 5$ and $T = 100$, there are on the order of $2 \cdot 100 \cdot 5^{100} \approx 10^{72}$ computations required. However, the forward-backward procedure, which is discussed below, offers an efficient way of calculating $P(O|\lambda)$.

2.4.6 Forward-Backward Procedure Both the forward calculation and backward calculation of the forward-backward procedure avoid the computational problems of (2.52) by using an inductive approach which averts the need for calculating $P(O, Q^i|\lambda)$ for all N^T possible state sequences. Derivation of the forward and backward algorithms are given is Appendix A.

2.4.6.1 Forward Algorithm The forward variable is defined as

$$\alpha_{t_n}(i) = P\left(O_{t_1} O_{t_2} \cdots O_{t_n}, q_{t_n} = S_i|\lambda\right) \tag{2.53}$$

or the joint probability of the partial observation sequence $O_{t_1} O_{t_2} \cdots O_{t_n}$ and being in state S_i at time t_n, given the model λ (21) (22). $\alpha_{t_n}(i)$ is evaluated inductively as follows:

1. Initialization:

$$\alpha_{t_1}(i) \;=\; b_i(O_{t_1})\pi_i, \quad 1 \le i \le N \tag{2.54}$$

2. Induction: Given $\alpha_{t_n}(i)$ for $1 \le i \le N$ calculate $\alpha_{t_{n+1}}(j)$ by induction:

$$\alpha_{t_{n+1}}(j) = b_j\left(O_{t_{n+1}}\right) \sum_{i=1}^{N} a_{ij}\alpha_{t_n}(i), \quad \begin{array}{l} 1 \le i \le N \\ 1 \le n \le T-1 \end{array} \tag{2.55}$$

   An illustration of the steps required to compute the forward variable $\alpha_{t_{n+1}}(j)$ is shown in Figure 2.2.

3. Termination: In final step, the desired probability, $P(O|\lambda)$, is computed.

$$P(O|\lambda) \;=\; \sum_{i=1}^{N} \alpha_{t_T}(i) \tag{2.56}$$

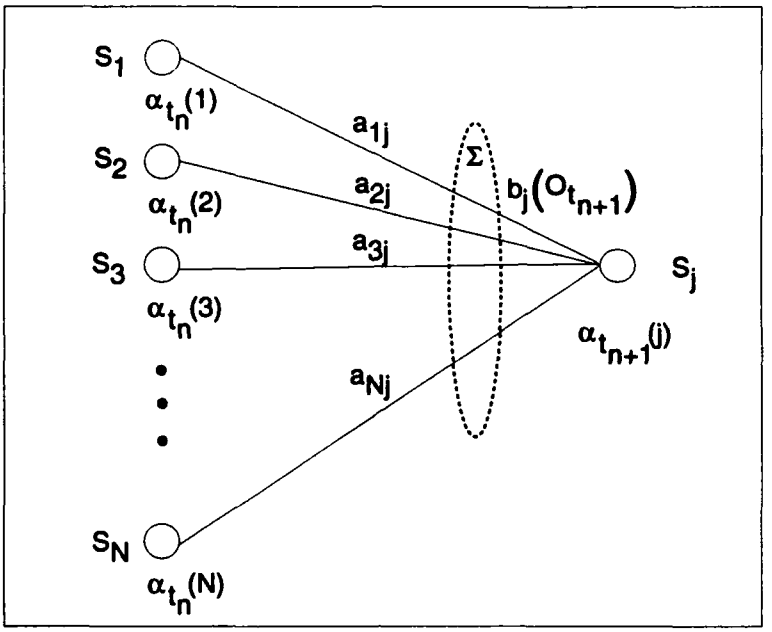


Figure 2.2. Calculation of the Forward Variable $\alpha_{t_{n+1}}(j)$

By (2.54), the number of computations required to compute the α's at t_1 is N multiplications, and referring to Figure 2.2 and (2.55), each α for $t_n > t_1$ requires $N+1$ multiplications and N additions for a total of $N \times T - 1$ α's. With the N additions in the termination step, the number of computation required to compute $P(O|\lambda)$, is $N + (N+1)(N)(T-1)$ multiplications and $N(N)(T-1) + N$ additions. Thus, the number of computations required is on the order of N^2T (about 3000 for $N = 5$ and $T = 100$) rather than the $2TN^T$ required by the direct calculation (21).

2.4.6.2 Backward Algorithm The backward algorithm also offers an efficient way of computing $P(O|\lambda)$. The derivation of the backward algorithm is also found in Appendix A. The backward variable $\beta_{t_n}(i)$ is defined as

$$\beta_{t_n}(i) = P\left(O_{t_{n+1}} O_{t_{n+2}} \cdots O_{t_T} | q_{t_n} = S_i, \lambda\right). \tag{2.57}$$

Hence, $\beta_{t_n}(i)$ is the probability of observing the partial observation sequence $O_{t_{n+1}} O_{t_{n+2}} \cdots O_{t_T}$, given state S_i and time t_n and the model λ (21) (22). Again $\beta_{t_n}(i)$ is calculated inductively as follows:

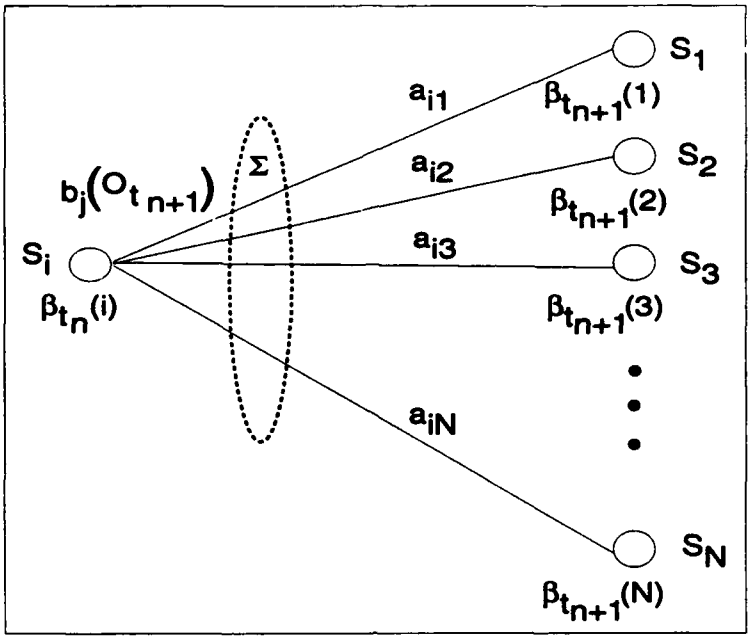


Figure 2.3. Calculation of the Backward Variable $\beta_{t_n}(i)$

1. Initialization: $\beta_{t_T}(i)$ is initialized to 1 for all i to maintain the desired probability. $\boldsymbol{O}$ is not explicitly defined for $t_n > t_T$, so the following probability is arbitrarily set to 1 (22).

$$\beta_{t_T}(i) \;=\; P\left(O_{t_T} \cdots O_{t_{T+1}} | q_{t_T} = S_i, \boldsymbol{\lambda}\right) \;=\; 1, \quad 1 \le i \le N \tag{2.58}$$

2. Induction: Compute β_{t_n} inductively using $\beta_{t_{n+1}}(i)$.

$$\begin{aligned} \beta_{t_n}(i) \;&=\; \textstyle\sum_{j=1}^{N} a_{ij} b_j\left(O_{t_{n+1}}\right) \beta_{t_{n+1}}(j), \quad T-1 \ge n \ge 1, \\ &\qquad\qquad 1 \le i \le N. \end{aligned} \tag{2.59}$$

The backward calculation is illustrated in Figure 2.3.

3. Termination: The desired probability is computed as

$$P\left(O_{t_1} \cdots O_{t_T} | \boldsymbol{\lambda}\right) \;=\; \sum_{i=1}^{N} \pi_i b_i\left(O_{t_1}\right) \beta_{t_1}(i) \tag{2.60}$$

The number of calculations required for the backward algorithm is approximately the same as the forward algorithm, so the forward algorithm holds no clear advantage over the backward algorithm. However, both are required for the solution of Problem 3.

2.4.7 Best State Sequence Given the Observation and the Model While Problem 1 has an exact solution, there are several possible solutions to Problem 2 which depend on the *optimality* criterion for the best state sequence Q given the observation sequence O and the model λ. One such *optimality* criterion is to choose the states q_{t_n} which are individually most likely at time t_n (22). The solution for this criteria is accomplished via the following variable which is defined as

$$\gamma_{t_n}(i) = P(q_{t_n} = S_i | O, \lambda) \tag{2.61}$$

or the probability of being in state S_i at time t_n, given the observation sequence O and the model λ. $\gamma_{t_n}(i)$ can be written in term of $\alpha_{t_n}(i)$ and $\beta_{t_n}(i)$ as shown by the derivation in Appendix A. $\gamma_{t_n}(i)$ is expressed in terms of $\alpha_{t_n}(i)$ and $\beta_{t_n}(i)$ as

$$\gamma_{t_n}(i) = \frac{\alpha_{t_n}(i)\beta_{t_n}(i)}{P(O|\lambda)} \tag{2.62}$$

The individually most likely state q_{t_n} at time t_n is solved in terms of $\gamma_{t_n}(i)$ as

$$q_{t_n} = \underset{1 \le i \le N}{\operatorname{argmax}} [\gamma_{t_n}(i)], \quad 1 \le n \le T. \tag{2.63}$$

Equation (2.63) maximizes the expected number of correct states, but the resulting state sequence may not be valid. For example, using (2.63) it is possible to choose $q_{t_n} = S_3$ and $q_{t_{n+1}} = S_2$ to be the most likely states at t_n and t_{n+1} even though a_{32} may be 0. Because the *optimal* state sequence defined by (2.63) may not be a valid sequence, it is not used for most applications (22). The most widely used optimality criterion is one which finds the single best state sequence (i.e. the sequence with the highest probability, given the observation and the model) so that $P(Q|O, \lambda)$ is maximized (22). Maximizing $P(Q|O, \lambda)$ also maximizes $P(Q, O|\lambda)$. The Viterbi algorithm provides an efficient technique for finding the single best state sequence.

2.4.8 Viterbi Algorithm Using the Viterbi algorithm, the single best state sequence is found via the quantity

$$\delta_{t_{n+1}}(j) = \max_{q_{t_1} q_{t_2} \cdots q_{t_n}} \left[P\left(q_{t_1} q_{t_2} \cdots q_{t_n}, q_{t_{n+1}} = S_j, O_{t_1} O_{t_2} \cdots O_{t_{n+1}} | \boldsymbol{\lambda}\right) \right]. \quad (2.64)$$

Where $\delta_{t_{n+1}}(j)$ is the probability of the best (or most likely) state sequence $q_{t_1} q_{t_2} \cdots q_{t_n}$ which also passes through S_j at t_{n+1} (or $q_{t_{n+1}} = S_j$) for the observation sequence, $O_{t_1} \cdots O_{t_{n+1}}$, given $\boldsymbol{\lambda}$. Solving for $\delta_{t_{n+1}}(j)$ directly will eventually incur the same computational difficulties as calculating $P(\boldsymbol{O}|\boldsymbol{\lambda})$ directly because there are N^{t_n} possible state sequences to test. However, if $\delta_{t_{n+1}}(j)$ is evaluated recursively in terms of $\delta_{t_n}(i)$ for $1 \leq i \leq N$, the probability of only N possible state sequences need to be scored because $\delta_{t_n}(i)$, by definition, is the probability of the most likely state sequence prior to t_n which passes through S_i for $1 \leq i \leq N$. The recursive relationship between $\delta_{t_{n+1}}(j)$ and $\delta_{t_n}(i)$ leads to the following expression,

$$\delta_{t_{n+1}}(j) = \max_i \left[\delta_{t_n}(i) a_{ij}\right] b_j\left(O_{t_{n+1}}\right) \quad (2.65)$$

The actual state sequence is tracked by recording the argument, $1 \leq i \leq N$, (which is actually the previous state) that maximizes (2.65). This value is denoted as the variable $\psi_{t_{n+1}}(j)$. The procedure is summarized below (22):

1. Initialization: The initialization step at t_1 does not require a scoring of the previous state sequence because the states prior to t_1 are not defined. Therefore, $\delta_{t_1}(i) = P\left(q_{t_1} = S_i, O_{t_1} | \boldsymbol{\lambda}\right)$ only depends on $\boldsymbol{\pi}$, $\boldsymbol{B}$, and O_{t_1}. $\psi_{t_1}(i)$ is not defined and is arbitrarily set to 0. The initialization step is as follows:

$$\begin{aligned} \delta_{t_1}(i) &= \pi_i b_i\left(O_{t_1}\right) \quad 1 \leq i \leq N \\ \psi_{t_1}(i) &= 0 \qquad\qquad 1 \leq i \leq N \end{aligned} \quad (2.66)$$

2. Induction: With initialization at t_1 complete, the recursive part of the Viterbi algorithm can proceed.

$$\delta_{t_{n+1}}(j) = \max_{1 \le i \le N} [\delta_{t_n}(i)a_{ij}]\, b_j\,(O_{t_{n+1}})\,, \quad 2 \le n \le T, \quad 1 \le j \le N \tag{2.67}$$

$$\psi_{t_{n+1}}(j) = \operatorname*{argmax}_{1 \le i \le N} [\delta_{t_n}(i)a_{ij}]\,, \quad 2 \le n \le T, \quad 1 \le j \le N \tag{2.68}$$

3. Termination: Once the probabilities of the best state sequences up to t_{T-1} have been determined the probability of the final best state sequence, denoted P^*, is simply the maximum $\delta_{t_T}(j)$ for $1 \le j \le N$ and the last state of the sequence is the argument of the maximum $\delta_{t_T}(j)$. Hence,

$$P^* = \max_{1 \le i \le N} [\delta_{t_T}(i)] \tag{2.69}$$

$$q^*_{t_T} = \operatorname*{argmax}_{1 \le i \le N} [\delta_{t_T}(i)]\,. \tag{2.70}$$

$$\tag{2.71}$$

4. State sequence retrieval: The most likely state sequence, $\boldsymbol{Q}^*$, is found by backtracking through the ψ's in the following way:

$$q^*_{t_n} = \psi_{t_{n+1}}\left(q^*_{t_{n+1}}\right) \quad n = T-1, T-2, \cdots 1 \tag{2.72}$$

2.4.9 HMM Synthesis through Training The third problem which is to adjust the parameters of the HMM model λ to maximize the probability of the observation sequence given the model does not have an analytical solution (21) (22). The Baum-Welch iterative estimation procedure does, however, provide a method which adjusts the model parameters so that $P(O\,|\lambda\,)$ is locally maximized. This procedure is presented below.

2.4.10 Baum-Welch Reestimation Procedure The Baum-Welch reestimation procedure uses two quantities. The first quantity, $\gamma_{t_n}(i)$, has been previously introduced and is defined in (2.61) and is the probability of being in S_i at t_n, given O and λ. The second quantity, $\xi_{t_n}(i,j)$, is defined as the probability of being in S_i at t_n and S_j at t_{n+1}, given O and λ such that (21) (22)

$$\xi_{t_n}(i,j) = P(q_{t_n} = S_i, q_{t_{n+1}} = S_j | O, \lambda) \tag{2.73}$$

$\xi_{t_n}(i,j)$ is also be expressed in terms of the forward and backward variables (α and β) as shown in Appendix A. $\xi_{t_n}(i,j)$ in terms of α and β is

$$\xi_{t_n}(i,j) = \frac{\alpha_{t_n}(i) a_{ij} b_j(O_{t_{n+1}}) \beta_{t_{n+1}}(j)}{P(O|\lambda)}, \quad 1 \le i \le N, \quad 1 \le j \le N \tag{2.74}$$

The definitions of $\xi_{t_n}(i,j)$ and $\gamma_{t_n}(i)$ are very similar. In fact, summing $\xi_{t_n}(i,j)$ over all j yields the probability of all transitions from S_i at t_n or equivalently the probability of being in S_i at t_n (21) (22). Therefore,

$$\gamma_{t_n}(i) = \sum_{j=1}^{N} \xi_{t_n}(i,j). \tag{2.75}$$

Another quantity of interest is a value which is proportional to the expected number of times that S_i is visited over the length of the observation sequence. This quantity is found by summing $\gamma_{t_n}(i)$ over all time $t_1 \le t_n \le t_T$ (22). Furthermore, if t_T is excluded from the summation, this quantity can then be interpreted as being proportional to the expected number of transitions from S_i during the observation sequence. Similarly, a quantity proportional to the expected number of transitions from S_i to S_j is the summation $\xi_{t_n}(i,j)$ for $t_1 \le t_n \le t_T$.

The Baum-Welch reestimation procedure is based upon the expected value of event occurrences or, in other words, the concept of counting event occurrences. The quantities described above, which are actually probability measures, provide a way to reestimate the HMM probability parameters. Accordingly, the Baum-Welch reestimation procedure is (21)(22):

1. Update the estimates of the initial state probabilities, π_i, to the probability proportional the expected number of times in S_i at t_1. The stochastic constraint, $\sum_{i=1}^{N} \overline{\pi}_i = 1$, is satisfied by

normalizing $\overline{\pi}_i$ by the probability which is proportional to the expected number of times in all states at t_1. $\gamma_{t_1}(i)$ satisfies this relationship so that $\overline{\pi}_i$ is estimated as

$$\overline{\pi}_i = \gamma_{t_1}(i) \tag{2.76}$$

$$= \frac{\alpha_{t_1}(i)\beta_{t_1}(i)}{\sum_{j=1}^{N} \alpha_{t_1}(j)\beta_{t_1}(j)} \tag{2.77}$$

for $1 \leq i \leq N$.

2. Update the estimates of the state transition probabilities, a_{ij}, to the probability proportional to the expected number of transitions from S_i to S_j for $t_1 \leq t_n \leq t_{T-1}$. The stochastic constraint, $\sum_{j=1}^{N} \overline{a}_{ij} = 1$, is satisfied in this case by normalizing $\overline{a}_{ij}$ by the probability which is proportional to the expected total number of transitions from S_i.

$$\overline{a}_{ij} = \frac{\sum_{n=1}^{T-1} \xi_{t_n}(i,j)}{\sum_{n=1}^{T-1} \gamma_{t_n}(i)} \tag{2.78}$$

$$= \frac{\sum_{n=1}^{T-1} \alpha_{t_n}(i) a_{ij} b_j\left(O_{t_{n+1}}\right) \beta_{t_{n+1}}(j)}{\sum_{n=1}^{T-1} \alpha_{t_n}(i)\beta_{t_n}(i)} \tag{2.79}$$

for $1 \leq i \leq N$ and $1 \leq j \leq N$.

3. Update the estimates of the state symbol probabilities, $b_i(m)$, to the probability proportional to the expected number of times in S_i and observing $O_{t_n} = v_m$ for $t_1 \leq t_n \leq t_T$. $\overline{b}_i(m)$ is normalized by dividing by the probability which proportional to the expected total number of times in S_i.

$$\overline{b}_i(m) = \frac{\sum_{n=1}^{T} \underbrace{\alpha_{t_n}(i)\beta_{t_n}(i)}_{\text{only if } O_{t_n} = v_m}}{\sum_{n=1}^{T} \alpha_{t_n}(i)\beta_{t_n}(i)} \tag{2.80}$$

for $1 \leq i \leq N$ and $1 \leq m \leq M$.

2.4.11 Properties of the Baum-Welch Reestimation Procedure Baum has proven that the reestimated model, $\overline{\lambda} = \left(\overline{A}, \overline{B}, \overline{\pi}\right)$ as defined by Equations (2.76) – (2.80), is either at a critical point, in which case $\overline{\lambda} = \lambda$, or $\overline{\lambda}$ is more likely than λ in the sense that $P(O|\overline{\lambda}) > P(O|\lambda)$ (7). Hence, the Baum-Welch algorithm insures that $P(O|\overline{\lambda}) \geq P(O|\lambda)$. By iteratively using $\overline{\lambda}$ in place of λ, the probability of O given the model is improved until the critical point is reached. The Baum-

Welch reestimation procedure leads to a local maximum only. For most problems, the optimization surface is very complex and has many local maxima (21) (22). Consequently, the critical point when $P(O|\overline{\lambda}) = P(O|\lambda)$, where λ represents the model at the end of the previous training loop, may not be the best possible model and is dependent on the starting point of the model before training (22). Methods for choosing the starting model are discussed in detail in references (11), (21), and (22).

2.4.12 Left-Right HMM Implementation For IWR a left-right HMM is usually implemented because the temporal relationship between phonemes is inherently modeled by the state transitions of the HMM (21) (22). A left-right HMM, which is shown in Figure 2.4, is a HMM which does not allow state transitions to a lesser state. Also, the initial state is always S_1. The initial state matrix π and the state transition matrix A for a 5 state left-right HMM are

$$\pi = \begin{pmatrix} 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{pmatrix} \tag{2.81}$$

and

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ 0.0 & a_{22} & a_{23} & a_{24} & a_{25} \\ 0.0 & 0.0 & a_{33} & a_{34} & a_{35} \\ 0.0 & 0.0 & 0.0 & a_{44} & a_{45} \\ 0.0 & 0.0 & 0.0 & 0.0 & a_{55} \end{pmatrix}. \tag{2.82}$$

Before implementing a left-right HMM, two problems must be resolved. The first problem is not unique to left-right HMM's and is directly related to the dynamic range of the computer used to implement the HMM's. To illustrate this problem, consider the forward calculation of $\alpha_{t_n}(i)$ defined by Equation (2.53). It can be shown the $\alpha_{t_n}(i)$ is calculated as the product of a large number of terms

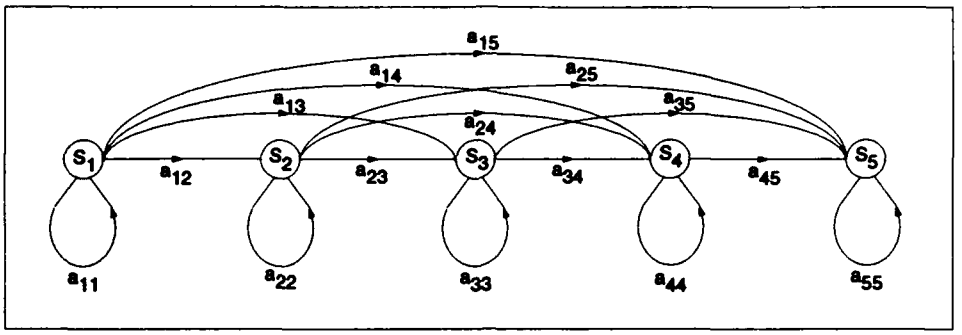


Figure 2.4. 5 State Left–Right HMM

that are significantly less than 1. Each calculation has the form (22)

$$\left(\prod_{s=1}^{n-1} a_{q_s q_{s+1}} \prod_{s=1}^{n} b_{q_{t_s}}(O_{t_s})\right).$$

The $\alpha_{t_n}(i)$'s exponentially approach zero as t_n increases. For t_n greater than 10, the computation of $\alpha_{t_n}(i)$ may exceed the dynamic range of the computer and for larger values of t_n, $\alpha_{t_n}(i)$ most certainly will underflow any computer. To avoid the underflow problem, both the $\alpha_{t_n}(i)$'s and the $\beta_{t_n}(i)$'s are calculated using a scaling procedure (22). Using the same reasoning it can be shown that the Viterbi algorithm will also underflow the computer for approximately the same values of t_n. However, the underflow problem with the Viterbi algorithm can be effectively avoided using logarithms (22).

Secondly, the left-right HMM cannot effectively use a single observation sequence to train because the transient nature of the states within the model only allow a small number of observations for any state for a given observation sequence. Therefore, in order to have sufficient data for reliable estimates, multiple observation sequences must be used to train the models (22). Consequently, Equations (2.78) and (2.80) must be modified to account for multiple observations. The initial state matrix, π , is not changed because the model always starts in S_1. The forward-backward variable scaling, Viterbi algorithm with logarithms and multiple observation sequence training procedure are discussed below.

2.4.12.1 Forward-Backward Variable Scaling The goal of the scaling procedure is to keep the $\alpha_{t_n}(i)$ and $\beta_{t_n}(i)$ coefficients within the dynamic range of the computer. The basic procedure

is to multiply $\alpha_{t_n}(i)$ by a scaling factor, c_{t_n}, which depends only on t_n and is independent of i. With the scaling factor incorporated the forward calculation is

1. Initialization:

$$\alpha_{t_1}(i) = b_i(O_{t_n})\pi_i, \quad 1 \leq i \leq N \tag{2.83}$$

The scaling coefficient is calculated for t_1 as

$$c_{t_1} = \frac{1}{\sum_{i=1}^{N} \alpha_{t_1}(i)} \tag{2.84}$$

and $\hat{\alpha}_{t_1}(i)$ ($\alpha_{t_1}(i)$ after scaling) is

$$\hat{\alpha}_{t_1}(i) = c_{t_1}\alpha_{t_1}(i), \quad 1 \leq i \leq N. \tag{2.85}$$

2. Induction: Given $\hat{\alpha}_{t_n}(i)$ for $1 \leq i \leq N$ calculate $\hat{\alpha}_{t_{n+1}}(j)$ by induction:

$$\tilde{\alpha}_{t_{n+1}}(j) = b_j(O_{t_{n+1}}) \sum_{i=1}^{N} a_{ij}\hat{\alpha}_{t_n}(i), \quad 1 \leq j \leq N \tag{2.86}$$

The scaling coefficient for t_{n+1} is calculated as

$$c_{t_{n+1}} = \frac{1}{\sum_{j=1}^{N} \tilde{\alpha}_{t_{n+1}}(j)} \tag{2.87}$$

and finally, $\hat{\alpha}_{t_{n+1}}(j)$ is

$$\hat{\alpha}_{t_{n+1}}(j) = c_{t_{n+1}}\tilde{\alpha}_{t_{n+1}}(j) \quad 1 \leq j \leq N. \tag{2.88}$$

This procedure is repeated recursively for $t_1 \leq t_n \leq t_{T-1}$.

3. Termination: $P(\boldsymbol{O}\,|\boldsymbol{\lambda}\,)$ cannot be calculated using $\hat{\alpha}_{t_T}(j)$ directly because the stochastic properties of the forward variable calculations have been corrupted by scaling. However, by induction

it can be shown that $\widehat{\alpha}_{t_n}(i)$, $\alpha_{t_n}(i)$, and c_{t_n} are related as follows (22).

$$\widehat{\alpha}_{t_{n+1}}(j) = \left(\prod_{s=t_1}^{t_{n+1}} c_s\right) \alpha^k_{t_{n+1}}(j) = C_{t_{n+1}} \alpha_{t_{n+1}}(j) \tag{2.89}$$

where $C_{t_{n+1}} = \prod_{s=t_1}^{t_{n+1}} c_s$. Hence, $P(O\,|\lambda)$ can be calculated in terms of C_{t_T}. This is shown by writing c_{t_T} as

$$\begin{aligned} c_{t_T} &= \frac{1}{\sum_{j=1}^{N} b_j(O_{t_{T-1}}) \sum_{i=1}^{N} a_{ij} \widehat{\alpha}_{t_{T-1}}(i)} \\ &= \frac{1}{C_{t_{T-1}} \sum_{j=1}^{N} b_j(O_{t_{T-1}}) \sum_{i=1}^{N} a_{ij} \alpha_{t_{T-1}}(i)} \\ &= \frac{1}{C_{t_{T-1}} \sum_{j=1}^{N} \alpha_{t_T}(j)}. \end{aligned} \tag{2.90}$$

It follows directly from (2.90) that

$$C_{t_T} \sum_{j=1}^{N} \alpha_{t_T}(j) = 1$$

and that

$$P(O\,|\lambda) = \frac{1}{C_{t_T}} = \frac{1}{\prod_{s=t_1}^{t_T} c_s} \tag{2.91}$$

If scaling is required, $P(O\,|\lambda)$ is also likely to be out of the dynamic range of the computer, so the log of $P(O\,|\lambda)$ is normally computed such that

$$\log[P(O\,|\lambda)] = -\sum_{s=t_1}^{t_T} \log[c_s] \tag{2.92}$$

The $\beta_{t_n}(i)$ coefficients are scaled by the same scaling factors for each time t_n as was used to scale the $\alpha_{t_n}(i)$. The backward recursive procedure is now

1. Initialization: $\widehat{\beta}_{t_T}(i)$ is initialized to c_{t_T} for all i.

$$\widehat{\beta}_{t_T}(i) = c_{t_T}, \quad 1 \le i \le N \tag{2.93}$$

2. Induction: Compute $\widehat{\beta}_{t_n}$ inductively using $\widehat{\beta}_{t_{n+1}}(i)$.

$$\begin{aligned} \widehat{\beta}_{t_n}(i) &= c_{t_n} \sum_{j=1}^{N} a_{ij} b_j (O_{t_{n+1}}) \widehat{\beta}_{t_{n+1}}(j), \quad t_{T-1} \geq t_n \geq t_1, \\ & \qquad\qquad 1 \leq i \leq N. \end{aligned} \tag{2.94}$$

3. Termination: $P(O|\lambda)$ cannot be computed after this scaling procedure because in this case the scaling coefficients are not explicitly related to the unscaled $\beta_{t_n}(i)$ coefficients. However, the goal of this scaling procedure is to keep the $\widehat{\beta}_{t_n}(i)$ coefficients within the dynamic range of the computer so they can be effectively used for the Baum-Welch reestimation procedure.

2.4.12.2 Viterbi Algorithm with Logarithms For the Viterbi algorithm the underflow problem is solved without scaling by using logarithm addition in place of the multiplications in Equations (2.66) and (2.67). Thus, the Viterbi algorithm using logarithms is implemented as shown below.

1. Initialization:

$$\begin{aligned} \log(\delta_{t_1}(i)) &= \log(\pi_i) + \log(b_i(O_{t_1})) \quad 1 \leq i \leq N \\ \psi_{t_1}(i) &= 0 \qquad\qquad\qquad\qquad 1 \leq i \leq N \end{aligned} \tag{2.95}$$

2. Induction:

$$\log(\delta_{t_{n+1}}(j)) = \max_{1 \leq i \leq N} [\log(\delta_{t_n}(i)) + \log(a_{ij})] + \log(b_j(O_{t_{n+1}})), \tag{2.96}$$

$$\psi_{t_{n+1}}(j) = \operatorname*{argmax}_{1 \leq i \leq N} [\log(\delta_{t_n}(i)) + \log(a_{ij})], \tag{2.97}$$

for $t_2 \leq t_n \leq t_T$ and $1 \leq j \leq N$.

3. Termination:

$$\log(P^*) = \max_{1 \le i \le N} [\log(\delta_{t_T}(i))] \tag{2.98}$$

$$q^*_{t_T} = \operatorname*{argmax}_{1 \le i \le N} [\log(\delta_{t_T}(i))]. \tag{2.99}$$

4. State sequence retrieval:

$$q^*_{t_n} = \psi_{t_{n+1}}\left(q^*_{t_{n+1}}\right) \quad n = T-1, T-2, \cdots 1 \tag{2.100}$$

2.4.12.3 Training with Multiple Observation Sequences and Scaled Forward-Backward Variables Recall that the Baum-Welch reestimation procedure maximizes $P(\boldsymbol{O}^k|\boldsymbol{\lambda})$ where the $\boldsymbol{O}^k$ denotes the k^{th} observation sequence. Note, that for the reestimation Equations (2.78) and (2.80), a single observation sequence is implied. For multiple observation sequences, the goal is to estimate the model parameters to maximize $P(\boldsymbol{O}|\boldsymbol{\lambda})$ for the entire set of K observation sequences denoted as

$$\boldsymbol{O} = \left(\boldsymbol{O}^1 \;\; \boldsymbol{O}^2 \;\; \ldots \;\; \boldsymbol{O}^k \;\; \ldots \;\; \boldsymbol{O}^K\right) \tag{2.101}$$

where the k^{th} observation sequence is

$$\boldsymbol{O}^k = \left(O^k_{t_1} \;\; O^k_{t_2} \;\; \ldots \;\; O^k_{t_T}\right). \tag{2.102}$$

Assuming that the observation sequences are independent,

$$P(\boldsymbol{O}|\boldsymbol{\lambda}) = \prod_{k=1}^{K} P(\boldsymbol{O}^k|\boldsymbol{\lambda}). \tag{2.103}$$

However, the log of $P(\boldsymbol{O}\,|\boldsymbol{\lambda}\,)$ is normally computed because even for small values of K and T it is likely to be out of the dynamic range of the computer. Thus,

$$\log\left[P(\boldsymbol{O}\,|\boldsymbol{\lambda}\,)\right] \;=\; \sum_{k=1}^{K}\log\left[P(\boldsymbol{O}^{\,k}|\boldsymbol{\lambda}\,)\right] \tag{2.104}$$

For the remainder of this discussion, let $P(\boldsymbol{O}^{\,k}|\boldsymbol{\lambda}\,) \equiv P^k$.

As shown in Section 2.4.10, the estimate of the state transition probability, $\overline{a}_{ij}$, is a probability proportional to the expected number of transitions from S_i to S_j for $t_1 \le t_n \le t_{T-1}$ normalized by the probability proportional to the expected total number transitions from S_i. For multiple observations, $\overline{a}_{ij}$ is a probability proportional to the sum total of the expected number of transitions from S_i to S_j for $t_1 \le t_n \le t_{T-1}$ over the entire set of K observation sequences normalized by the probability proportional to the expected total number transitions from S_i over the K observation sequences. Hence, the modified reestimation formula for $\overline{a}_{ij}$ is

$$\begin{aligned}\overline{a}_{ij} \;&=\; \frac{\sum_{k=1}^{K}\sum_{n=1}^{T-1}\xi_{t_n}^{k}(i,j)}{\sum_{k=1}^{K}\sum_{n=1}^{T-1}\gamma_{t_n}^{k}(i)}\\ &=\; \frac{\sum_{k=1}^{K}\frac{1}{P^k}\sum_{n=1}^{T-1}\alpha_{t_n}^{k}(i)a_{ij}b_j\left(O_{t_{n+1}}^{k}\right)\beta_{t_{n+1}}^{k}(j)}{\sum_{k=1}^{K}\frac{1}{P^k}\sum_{n=1}^{T-1}\alpha_{t_n}^{k}(i)\beta_{t_n}^{k}(i)}\end{aligned} \tag{2.105}$$

for $1 \le i \le N$ and $1 \le j \le N$. Similarly, the modified reestimation formula for $\overline{b}_i(m)$ is

$$\begin{aligned}\overline{b}_i(m) \;&=\; \frac{\sum_{k=1}^{K}\sum_{n=1}^{T}\underbrace{\gamma_{t_n}^{k}(i)}_{\text{only if } O_{t_n}^{k}\,=\,v_m}}{\sum_{k=1}^{K}\sum_{n=1}^{T}\gamma_{t_n}^{k}(i)}\\ &=\; \frac{\sum_{k=1}^{K}\frac{1}{P^k}\sum_{n=1}^{T}\underbrace{\alpha_{t_n}^{k}(i)\beta_{t_n}^{k}(i)}_{\text{only if } O_{t_n}^{k}\,=\,v_m}}{\sum_{k=1}^{K}\frac{1}{P^k}\sum_{n=1}^{T}\alpha_{t_n}^{k}(i)\beta_{t_n}^{k}(i)}\end{aligned} \tag{2.106}$$

for $1 \le i \le N$ and $1 \le m \le M$.

Each observation sequence, k, has its own set of $\alpha_{t_n}^{k}(i)$ and $\beta_{t_n}^{k}(i)$ coefficients as denoted by the superscript k in Equations (2.105) and (2.106). The $1/P^k$ factor in Equations (2.105) and (2.106) is present because it is part of the definitions of $\gamma_{t_n}^{k}(i)$ and $\xi_{t_n}^{k}(i,j)$ and does not cancel because those quantities are summed over k in the numerator and denominator. $P(\boldsymbol{O}\,|\boldsymbol{\lambda}\,)$ cancels in Equations (2.78)

and (2.80). If the $\alpha_{t_n}^k(i)$ and $\beta_{t_n}^k(i)$ coefficients have been scaled, according to Rabiner the reestimation equations become

$$\overline{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P^k} \sum_{n=1}^{T-1} \widehat{\alpha}_{t_n}^k(i) a_{ij} b_j \left(O_{t_{n+1}}^k\right) \widehat{\beta}_{t_{n+1}}^k(j)}{\sum_{k=1}^{K} \frac{1}{P^k} \sum_{n=1}^{T-1} \frac{1}{c_{t_n}^k} \left[\widehat{\alpha}_{t_n}^k(i) \widehat{\beta}_{t_n}^k(i)\right]} \tag{2.107}$$

for $1 \leq i \leq N$ and $1 \leq j \leq N$ and

$$\overline{b}_i(m) = \frac{\sum_{k=1}^{K} \frac{1}{P^k} \sum_{n=1}^{T} \underbrace{\widehat{\alpha}_{t_n}^k(i) \widehat{\beta}_{t_n}^k(i)}_{\text{only if } O_{t_n}^k = v_m}}{\sum_{k=1}^{K} \frac{1}{P^k} \sum_{n=1}^{T} \widehat{\alpha}_{t_n}^k(i) \widehat{\beta}_{t_n}^k(i)} \tag{2.108}$$

for $1 \leq i \leq N$ and $1 \leq m \leq M$. The $1/c_{t_n}^k$ factor in the denominator of Equation (2.107) is required because that scaling coefficient is *absent* from the numerator. The relationship between $\widehat{\alpha}_{t_n}^k(i)$, $\alpha_{t_n}^k(i)$, and $C_{t_n}^k$ is given by Equation (2.89) and a similar relationship also holds between the scaled and unscaled $\beta_{t_n}^k(i)$ coefficients. Thus,

$$\begin{aligned} \widehat{\alpha}_{t_n}^k(i) a_{ij} b_j \left(O_{t_{n+1}}^k\right) \widehat{\beta}_{t_{n+1}}^k(j) &= \left(\prod_{s=t_1}^{t_n} c_s^k\right) \alpha_{t_n}^k(i) a_{ij} b_j \left(O_{t_{n+1}}^k\right) \left(\prod_{s=t_{n+1}}^{T} c_s^k\right) \beta_{t_{n+1}}^k(j) \\ &= C_{t_T}^k \alpha_{t_n}^k(i) a_{ij} b_j \left(O_{t_{n+1}}^k\right) \beta_{t_{n+1}}^k(j) \end{aligned} \tag{2.109}$$

Therefore, the effect of the scaling coefficients can clearly be seen if Equations (2.107) and (2.108) are written as

$$\overline{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{C_{t_T}^k}{P^k} \sum_{n=1}^{T-1} \alpha_{t_n}^k(i) a_{ij} b_j \left(O_{t_{n+1}}^k\right) \beta_{t_{n+1}}^k(j)}{\sum_{k=1}^{K} \frac{C_{t_T}^k}{P^k} \sum_{n=1}^{T-1} \frac{c_{t_n}^k}{c_{t_n}^k} \left[\alpha_{t_n}^k(i) \beta_{t_n}^k(i)\right]} \tag{2.110}$$

for $1 \leq i \leq N$ and $1 \leq j \leq N$ and

$$\overline{b}_i(m) = \frac{\sum_{k=1}^{K} \frac{C_{t_T}^k}{P^k} \sum_{n=1}^{T} \underbrace{c_{t_n}^k \alpha_{t_n}^k(i) \beta_{t_n}^k(i)}_{\text{only if } O_{t_n}^k = v_m}}{\sum_{k=1}^{K} \frac{C_{t_T}^k}{P^k} \sum_{n=1}^{T} c_{t_n}^k \alpha_{t_n}^k(i) \beta_{t_n}^k(i)} \tag{2.111}$$

According to Rabiner, dividing by P^k removes the scaling factor from each term before summing over k because of the fact that $P^k = 1/C_{t_T}^k$(22). However, Rabiner does not address how Equations (2.107)

and (2.108) would be implemented if P^k is out the dynamic range of the computer – the primary reason for scaling in the first place. These issues are addressed in Section 3.5.1.

2.5 Classification Performance Clarification

In Chapter IV, estimates of the classification error performance will be reported for the various classifiers at SNR's of 20 dB, 15 dB, 10 dB, 5 dB, 0 dB, and -5 dB. To present these estimates in some meaningful way, the SNR will be related to range and the accuracy of the estimates will be specified. Hence, the purpose of this section is to clarify the results of the classification experiments to follow.

2.5.1 Relating SNR to Range The SNR can easily be changed in computer simulations but, it may not be the most useful quantity for expressing the overall performance of the RTI algorithm under test. Certainly from the pilot's perspective, a more useful quantity is the range to the target, which is inversely proportional to the SNR. Actually, the SNR for a particular range is dependent on a number of variables. One such variable is the radar cross section (RCS) of the target. However, the RCS is a volatile quantity which is very dependent on aspect angle; changing by as much as 30 dB for just 0.2° change in aspect angle (4). Obviously, the SNR for a given range is also dependent on the radar system parameters. In this section, the radar range equation (RRE) will be used to compute the range given the SNR (26).

A hypothetical HRR linear FM pulse compression radar, whose transmitted signal is same as described in Sections 2.2.2 and 3.2.3, and a hypothetical target will be used. The RRE for this hypothetical radar and target is

$$R^4 = \frac{P_{av} G A \sigma}{(4\pi)^2 k T_o F_s (B_{fm} T) f_p \text{SNR}} \tag{2.112}$$

where $R \equiv$ range from the radar to the target, m

$P_{av} \equiv$ average transmitted power = 34 w

$G \equiv$ antenna gain = 10,000

$A \equiv$ antenna effective aperture = 1.5 m^2

$\sigma \equiv$ RCS of target = 2 m^2

$k \equiv$ Boltzmann's constant = 1.38×10^{-23} J/deg

$T_o \equiv$ standard temperature = 290 K

$F_s \equiv$ system noise figure = 2

$B_{fm} \equiv$ receiver bandwidth = 750 MHz

$T \equiv$ pulse width = 170.67 ns

$f_p \equiv$ pulse repetition frequency = 1000 Hz

SNR $\equiv$ Signal to noise ratio

Although, the parameters of the radar are carefully chosen so that reasonable ranges for a given SNR can be realized, no attempt is made to insure that the radar can be made with todays technology. For example, an average power of 34 w for a pulse width of 170.67 ns and a pulse repetition frequency of 1000 Hz, requires a peak power of 200 kw. At the very least, it would be difficult to fit an aircraft with a transmitter this powerful. To field a system with an antenna gain of 10,000 and system noise figure of 2 is equally as difficult. To further complicate the issue, the RCS of the target fluctuates wildly so that the range for given the SNR will also fluctuate. With all that, the range for a given SNR is given in Table 2.1.

Table 2.1. Range versus SNR

SNR	20 dB	15 dB	10 dB	5 dB	0 dB	- 5 dB
Range	15.8 km	21.1 km	28.2 km	37.6 km	50.1 km	79.5 km

2.5.2 Determining Error Performance Confidence Intervals The classification results presented in Chapter IV are based on a finite data set and are, therefore, only point estimates of the actual classification rate, E (24). Let the estimate of E be defined as

$$\widehat{E} \equiv \frac{L}{P}$$

$L \equiv$ Number of misclassified test vectors

$P \equiv$ Number of test vectors

Assuming that $\widehat{E}$ is a binomial random variable, the *Classifier Confidence Interval* is related to the variance of $\widehat{E}$ and E is the mean of $\widehat{E}$ (24). By expressing the distribution of $\widehat{E}$ as the Gaussian

approximation to the binomial distribution, the variance of $\hat{E}$ is approximated by $\hat{E}(1-\hat{E})/P$. It can be shown that

$$Prob\left\{\hat{E} - z\hat{\sigma}\{\hat{E}\} < E < \hat{E} - z\hat{\sigma}\{\hat{E}\}\right\} \approx \gamma \tag{2.113}$$

$$\hat{\sigma}\{\hat{E}\} \equiv \sqrt{\frac{\hat{E}(1-\hat{E})}{P}} \tag{2.114}$$

where z and γ are parameters of the standard Gaussian distribution of $\hat{E}$.

Thus, the actual classification rate is within the confidence interval, $\hat{E} - z\hat{\sigma}\{\hat{E}\} < E > \hat{E} - z\hat{\sigma}\{\hat{E}\}$, with a confidence level or probability of γ. For example, if $z = 1.96$ the confidence level is 95% ($\gamma = 0.95$).

2.6 Summary

In this chapter, the background material for a linear FM pulse compression HRR radar, the Prony model, and HMM's has been presented. These three disciplines comprise the major components of the the RTI algorithms developed for this thesis. Also, the SNR was related to the range from the radar to the target and the accuracy of the classification rate estimates was defined to clarify the classification performance reported in Chapter IV. In the next chapter, the RTI algorithm will be described by drawing upon the background material presented here.

III. RTI ALGORITHM DESCRIPTION

3.1 Introduction

The purpose of this chapter is to describe the RTI algorithms developed for this research effort. The background material presented in Chapter II forms the basis for three major sub-processes of the basic RTI algorithm. These three major sub-processes are the front-end signal processing process, the feature extraction process, and the classification process. The system block diagram is shown in Figure 3.1. What follows will be a detailed description of each element within the block diagram.

3.2 Front-end Signal Processing

The purpose of the front-end signal processing is to condition the non-causal, noiseless Xpatch HRR radar signatures so that they emulate causal, and therefore physically realizable, circularly polarized, HRR Linear FM, noise corrupted radar signatures. This process is described below.

3.2.1 HRR Radar Signature Synthesis To reduce the data storage requirements, the raw Xpatch output contains only the positive (one-sided) spectrum of the band limited impulse response of the target. Thus, in software, Xpatch illuminates the target with a complex signal, which has a flat spectrum over the entire bandwidth of approximately 1.5 GHz. In the discrete time domain, where n is related to range r_n by Equation (2.6), the incident signal for a single linear polarization is

$$\begin{aligned} x[n] &= \frac{\sin(2\pi BT_s n)}{\pi n} e^{j2\pi f_c T_s n},\ n = 0, 1, 2, \ldots 1023 \qquad (3.1) \\ f_c &\approx 10.0 \text{ GHz} \\ B = \frac{1}{T_s} = f_s &\approx 1.5 \text{ GHz}. \end{aligned}$$

It should be noted that T_s is not an even multiple of f_c. As a result, the discrete spectrum of $x[n]$, which is periodic in multiples of 2π, is not centered around f_c. Although it is not certain how Xpatch resolves this issue, the discrete spectrum of $x[n]$ can easily be centered by circularly shifting the spectrum to the right about 250 MHz or 170 discrete frequency points. Thus, the incident signal in the discrete frequency domain is the shifted DFT of $x[n]$ or

$$X[k] = 1,\ k = 0, 1, 2, \cdots 1023 \qquad (3.2)$$

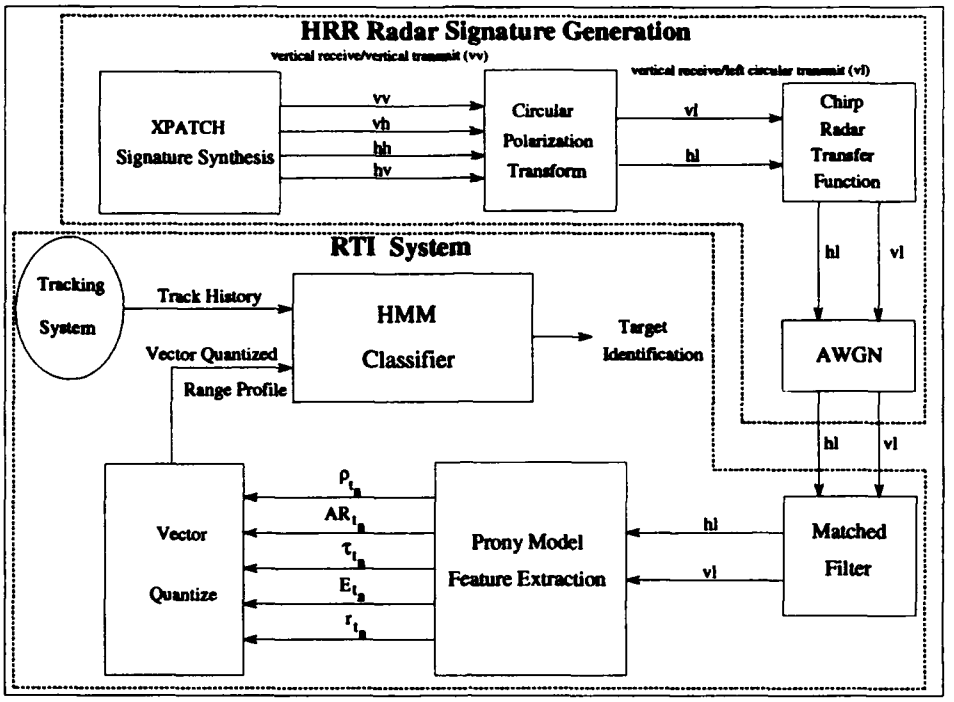


Figure 3.1. HRR Signature Generation and RTI System Block Diagram

$$k \equiv f_k - \left(f_c - \frac{B}{2}\right)\Delta f$$

$$\Delta f = \frac{f_s}{1024 - 1} \approx 1.467 \text{ MHz}$$

Xpatch creates a fully polarized signature or range profile by generating 4 signatures for each signature calculation: vertical receive/vertical incident (vv); vertical receive/horizontal incident (vh); horizontal receive/horizontal incident (hh); and horizontal receive/vertical incident (hv). The four signatures are represented in the frequency domain and consist of 1024 discrete frequency points as defined in Equation (3.2). By Equation (2.7) the range extent of the signature is 102.4 meters and by Equation (2.4) the range resolution is approximately 0.1 meters. The output data format of the Xpatch signatures is described in Appendix B.

3.2.2 Frequency Decimation and Linear to Circular Polarization Transformation As stated in Section 3.2.1, the range extent of the range profiles is 102.4 meters. However, the targets of interest are fighter class aircraft which are less than 25 meters long. Therefore, the output from Xpatch is decimated, keeping one of four frequency samples, to increase Δf by a factor of 4, thus reducing the range extent to 25.6 meters. Also, without any loss of information the four decimated linearly polarized Xpatch range profiles are transformed into two left circularly polarized range profiles by the transform given by Equation (2.11). Hence, the fully polarized range profiles are represented by vertical receive/left circular incident, ($X_{vl}[k]$), and horizontal receive/left circular incident, ($X_{hl}[k]$), range profiles where $0 \leq k \leq 255$.

3.2.3 Linear FM Pulse Compression At this point the range profile is essentially the impulse response of target over the finite bandwidth B and is free of any noise. A realistic response of the target is achieved by filtering the range profile by the 'chirped' HRR radar transfer function described in Section 2.2.2, adding white Gaussian noise, and then filtering with a transfer function that is matched to the original 'chirped' HRR radar transfer function. This transfer function is given by Equation (2.10). To correspond to the Xpatch output, the complex form of this equation in the discrete time domain is given here.

$$
\begin{aligned}
f[n] &= e^{j\left(2\pi f_c\left[nT_s-\frac{T}{2}\right]+\int \frac{2\pi B_{fm}}{T}\left[nT_s-\frac{T}{2}\right]dnT_s\right)}\,\mathrm{rect}\left(\frac{\left[n-\frac{T}{2T_s}\right]}{\frac{T}{T_s}}\right), \qquad (3.3)\\
n &= 0, 1, 2, \ldots 255\\
f_c &\equiv \text{10.0 GHz: carrier frequency}\\
T_s = \frac{1}{f_s} &\equiv \text{1.5 GHz: sampling frequency}\\
B_{fm} &\equiv \text{750 MHz: bandwidth}\\
T &\equiv \text{170.67 ns or 25.6 m: pulse width}\\
B_{fm}T &= \text{128: pulse compression ratio}
\end{aligned}
$$

Again the spectrum of $f[n]$, $F[k]$, will not be centered in the periodic spectral window, so $F[k]$ is circularly shifted to center the spectrum as illustrated in Figure 3.2. In the discrete frequency domain, the circularly polarized Xpatch signatures, $X_{vl}[k]$ and $X_{hl}[k]$, are multiplied by $F[k]$. After $X_{vl}[k]$ and $X_{hl}[k]$ are corrupted by AWGN, as describe below, they are both match filtered by $F^*[k]$.

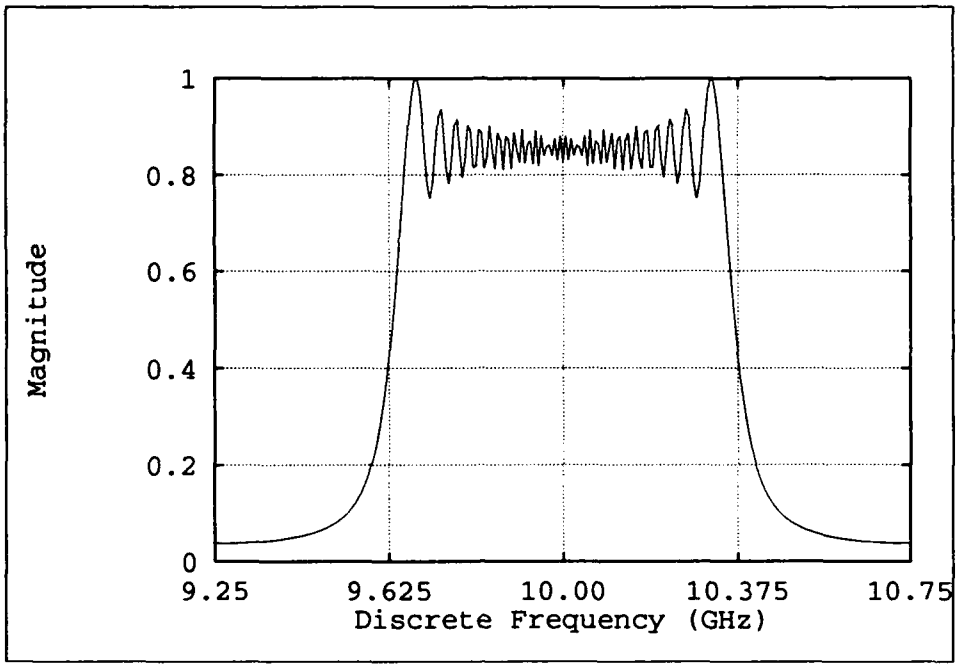


Figure 3.2. Magnitude Response of $F[k]$

3.2.4 Noise Corruption As stated in Chapter I, it is assumed that the target is in free space away from any clutter or other noise sources. Therefore, the primary sources of noise are assumed to be thermal noise received by the radar antenna and thermal noise in the receiver which is adequately modeled as AWGN (9). $X_{vl}[k]$ and $X_{hl}[k]$ are transformed to the discrete time domain with an IFFT routine, corrupted by complex (complex because the signals are complex) AWGN, and then transformed back to the discrete frequency domain using a FFT routine. $X_{vl}[k]$ and $X_{hl}[k]$ are corrupted by independent noise realizations because it is assumed that they are processed by a two channel receiver. To set the notation, the outputs of the matched filters, which now emulate noisy circularly polarized Linear FM HRR radar signatures, are $S_{vl}[k]$ and $S_{hl}[k]$.

3.3 Feature Extraction via the Prony Model

The scattering centers of the target are modeled using the Prony algorithm discussed in Section 2.3. Referring to Figure 3.2, the frequency samples outside bandwidth of $F[k]$ are essentially zero and the data set is further reduced to the frequency samples within the bandwidth of $F[k]$. The definition

for the Prony Model of fully polarized range profiles is restated here as

$$\begin{pmatrix} S_{hl}[k] \\ S_{vl}[k] \end{pmatrix} = \sum_{n=1}^{T} \begin{pmatrix} a_{ht_n} \\ a_{vt_n} \end{pmatrix} p_{t_n}^{k}, \quad k = 0, 1, 2, ..., N-1 \tag{3.4}$$

where

p_{t_n} ≡ pole of the n^{th} scattering center

a_{ht_n} ≡ amplitude coefficient of the horizontally received n^{th} scattering center

a_{vt_n} ≡ amplitude coefficient of the vertically received n^{th} scattering center

T ≡ number of scattering centers

N ≡ number of discrete frequencies.

Recall that the frequency responses of the scattering centers are represented by the Prony model, therefore N, the number of discrete frequencies, is limited to the 116 samples ($\approx$ 680 MHz) in the center of $F[k]$ where the frequency response is relatively flat. Each range profile is modeled using the T scattering centers with the most energy. Each scattering center along the range profile is characterized by 5 features as described in section 2.3.1. In matrix form each range profile is written as

$$\text{range profile} = \begin{pmatrix} |p_{t_1}| & AR_{t_1} & \tau_{t_1} & E_{t_1} & \tau_{t_1} \\ |p_{t_2}| & AR_{t_2} & \tau_{t_2} & E_{t_2} & \tau_{t_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ |p_{t_T}| & AR_{t_T} & \tau_{t_T} & E_{t_T} & \tau_{t_T} \end{pmatrix} \tag{3.5}$$

The parameter estimation procedure does not inherently order the scattering centers according range, so after the parameter estimation procedure the scattering centers are sorted a placed in ascending order according to range.

3.4 Vector Quantization

Depending on the classification algorithm, (these algorithms are described Section 3.5) the vector quantization process assigns a code book value which corresponds to a predetermined clustering center

within the 5 dimensional feature space created by the 5 parameters that define each scattering center or the $5 \times T$ dimensional feature space created by the entire set of T scattering centers of the range profile. The location of the clustering centers are found using the K-means clustering option in LNKnet which is a multi-purpose clustering and classification software package developed by MIT Lincoln Laboratories.

Specifically, the clustering centers are found by normalizing each column of Equation (3.5), with respect to the entire training data set, to zero mean and unit variance. The training data set contains range profiles from both targets at all aspect angles of interest. Once the clustering centers have been defined, each normalized scattering center is assigned the codebook value which corresponds to the closest clustering center with respect to Euclidian distance. Range profiles from the evaluation data set are normalized by the same mean and variance of the training data set and are vector quantized to the same clustering centers derived from the training data set.

If each scattering center of the k^{th} range profile is vector quantized, this range profile is represented by a sequence of T symbols (integers in this case) from the symbol set $V = \{v_1, v_2, \ldots, v_m, \ldots, v_M\}$, where M is the number of symbols. The k^{th} range profile is written as

$$O^k = O^k_{t_1} \quad O^k_{t_2} \quad \ldots \quad O^k_{t_n} \quad \ldots \quad O^k_{t_T}$$

where T is the number of scattering centers and $O_{t_n} = v_m$. Similarly, if the entire range profile is vector quantized to one code book symbol, a sequence of range profiles is represented in the same way except that T corresponds to the number of range profiles in the sequence and t_n is the n^{th} range profile.

3.5 Classification using HMM's

Referring to Figure 3.1 the HMM classification block has 2 inputs. One of these inputs is the track history from the tracking system and assumed to be accurate to within $\pm 5°$ (16). The other input is the vector quantized range profiles.

In general, the classification process is composed of two distinct procedures: the training procedure and the classification procedure. During the training procedure a HMM (or several HMM's depending the classifier implementation) for each target is synthesized from the training data set

using the Baum-Welch reestimation procedure (21) (22). Classification of unknown range profiles (or sequences of range profiles) is accomplished by calculating the probability of the unknown range profile given the HMM (or HMM's) from each class. The class with the highest probability is assigned to the unknown range profile (or sequences of range profiles).

3.5.1 HMM Implementation The temporal relationship between the scattering centers as well as the range profiles suggest that a left-right HMM is well suited for this problem. As stated in Section 2.4.12, a left-right HMM does not allow state transitions to a lesser state. Also, the initial state is always state 1. In Section 2.4.12.3, the Baum-Welch training procedure with multiple sequences and scaled forward-backward coefficients is discussed and the justification for why Equations (2.107) and (2.108) will not work is given. In this section, the training algorithm developed for this research effort will be described.

The modified reestimation equations used for this research effort are

$$\overline{a}_{ij} = \frac{\sum_{k=1}^{K} \sum_{n=1}^{T-1} \frac{1}{c_{t_n}^k} \widehat{\alpha}_{t_n}^k(i) a_{ij} b_j\left(O_{t_{n+1}}^k\right) \widehat{\beta}_{t_{n+1}}^k(j)}{\sum_{k=1}^{K} \sum_{n=1}^{T-1} \frac{1}{\left(c_{t_n}^k\right)^2} \left[\widehat{\alpha}_{t_n}^k(i) \widehat{\beta}_{t_n}^k(i)\right]} \tag{3.6}$$

for $1 \leq i \leq N$ and $1 \leq j \leq N$ and

$$\overline{b}_i(m) = \frac{\sum_{k=1}^{K} \sum_{n=1}^{T} \underbrace{\frac{1}{c_{t_n}^k} \widehat{\alpha}_{t_n}^k(i) \widehat{\beta}_{t_n}^k(i)}_{\text{only if } O_{t_n}^k = v_m}}{\sum_{k=1}^{K} \sum_{n=1}^{T} \frac{1}{c_{t_n}^k} \widehat{\alpha}_{t_n}^k(i) \widehat{\beta}_{t_n}^k(i)} \tag{3.7}$$

for $1 \leq i \leq N$ and $1 \leq m \leq M$. To show why Equations (3.6) and (3.7) are valid, first Equation (3.7) is expressed in terms of the unscaled $\alpha_{t_n}(i)$ and $\beta_{t_n}(i)$ coefficients as

$$\begin{aligned} \overline{b}_i(m) &= \frac{\sum_{k=1}^{K} C_T^k \sum_{n=1}^{T} \underbrace{\frac{c_{t_n}^k}{c_{t_n}^k} \alpha_{t_n}^k(i) \beta_{t_n}^k(i)}_{\text{only if } O_{t_n}^k = v_m}}{\sum_{k=1}^{K} C_T^k \sum_{n=1}^{T} \frac{c_{t_n}^k}{c_{t_n}^k} \alpha_{t_n}^k(i) \beta_{t_n}^k(i)} \\ &= \frac{\sum_{k=1}^{K} \sum_{n=1}^{T} \underbrace{\gamma_{t_n}^k(i)}_{\text{only if } O_{t_n}^k = v_m}}{\sum_{k=1}^{K} \sum_{n=1}^{T} \gamma_{t_n}^k(i)} \end{aligned} \tag{3.8}$$

for $1 \leq i \leq N$ and $1 \leq m \leq M$. Thus, (3.7) is equivalent to (2.106). Similarly, Equation (3.6) is

$$\overline{a}_{ij} = \frac{\sum_{k=1}^{K} \sum_{n=1}^{T-1} \frac{1}{c_{t_n}^k} \xi_{t_n}^k(i,j)}{\sum_{k=1}^{K} \sum_{n=1}^{T-1} \frac{1}{c_{t_n}^k} [\gamma_{t_n}^k(i)]} \tag{3.9}$$

for $1 \leq i \leq N$ and $1 \leq j \leq N$. The additional $1/c_{t_n}^k$ factor in Equation (3.9) does not appear in (2.105), but was emperically found to work well for training left-right HMM's. A purely heuristic argument to explain why (3.9) works is that the $1/c_{t_n}^k$ factor slightly enhances the more probable observation sequences because the $c_{t_n}^k$ factors are generally smaller for the more probable observation sequences. As a result, the state transition probabilities are driven by the most likely observation sequences. Because, (3.9) is not equivalent to (2.105) the training procedure does not insure that $P(O|\overline{\lambda}) \geq P(O|\lambda)$, but it causes just enough equivocation to prevent $P(O|\overline{\lambda})$ from settling at a less than optimum critical point. The training procedure developed for this thesis and the experimental results that validate this training procedure are found in Chapter IV.

3.5.2 Classification Based on a Single Range Profile The HMM classifier can be implemented to classify unknown targets based on a single range profile or on a sequence of range profiles. Classification using single profiles is desirable because classification is nearly instantaneous and it generally requires only a rough estimate of the aspect angle. Using this method several HMM's, one for each aspect angle sector, can be used to represent the target. For example, a HMM could be synthesized using range profiles of the target over a 10° by 10° sector at the nose of the target. When the tracking system shows the target in that sector the corresponding HMM's for each target for that sector are used to classify the unknown target. A block diagram of such a classification algorithm for a single sector is shown in Figure 3.3.

3.5.3 Classification Based on Single Looks at Multiple Range Profiles To identify the target after a sequence of range profiles is received and processed, the single look classifier shown in Figure 3.3 must be modified slightly so that it waits to make the classification decision after the entire range profile sequence has been processed. So, the classification decision is now made by comparing the sum of the individual range profile log probabilities for each single look HMM's.

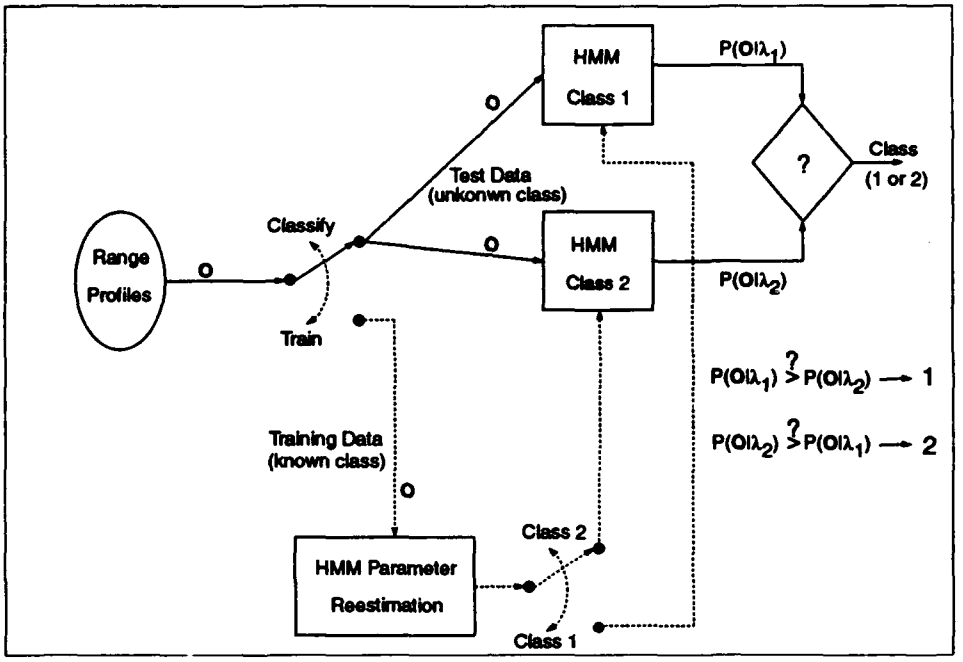


Figure 3.3. Single Range Profile HMM Classifier

3.5.4 Classification Based on Sequences of Vector Quantized Range Profiles Although classification based on a single range profile may be desirable, only part of the available information about the target is used. Assuming that the target makes a smooth transition through the aspect angle space, how the range profiles change from aspect angle to aspect angle also provide information that could be used for target classification (17).

One way to model the range profile changes is to reduce the entire range profile to one symbol via a vector quantization process. One HMM per aspect angle sector is synthesized for each target using sequences of range profiles (one integer per range profile). The classifier implementation for multiple range profiles is very similar to the single range profile classifier shown in Figure 3.3. The primary difference between the two is that the HMM is designed to model the change between sequences of range profiles (each range profile is represented by one symbol) instead of a sequence of scattering centers. Therefore, the classifier requires a number of range profiles to make the classification decision. Although the temporal relationships between the range profiles if used with this type of classifier, the

temporal relationships between the individual scattering centers along the range profiles are essentially lost by condensing all of the information of each range profile to a single number.

3.5.5 Classification Based on the State Transitions of Multiple Range Profiles Ideally, the classifier should model both the temporal relationships between the scattering centers as well as the temporal relationships between the range profiles. Figure 3.4 shows an implementation of such a classifier. The HMM's for each class on the left of the figure are Single Look HMM's in Figure 3.3. The outputs of the class 1 and class 2 single range profile HMM's are best state sequences of the individual range profiles as calculated by the Vertibi algorithm described in Sections 2.4.8 and 2.4.12.2. The remaining HMM's correspond to states of the single range profile HMM's. The state 1 HMM for class 1 is trained to model the scattering centers which belong to state 1 of the class 1 single profile HMM for the entire sequence of range profiles. The following example illustrates the process.

1. Let $\boldsymbol{O}$ be a sequence of 5 range profiles from an unknown target which is modeled as having 5 scattering centers. Hence,

$$
\begin{aligned}
\boldsymbol{O}^1 &= O^1_{t_1} \; O^1_{t_2} \; O^1_{t_3} \; O^1_{t_4} \; O^1_{t_5} \\
\boldsymbol{O}^2 &= O^2_{t_1} \; O^2_{t_2} \; O^2_{t_3} \; O^2_{t_4} \; O^2_{t_5} \\
\boldsymbol{O}^3 &= O^3_{t_1} \; O^3_{t_2} \; O^3_{t_3} \; O^3_{t_4} \; O^3_{t_5} \\
\boldsymbol{O}^4 &= O^4_{t_1} \; O^4_{t_2} \; O^4_{t_3} \; O^4_{t_4} \; O^4_{t_5} \\
\boldsymbol{O}^5 &= O^5_{t_1} \; O^5_{t_2} \; O^5_{t_3} \; O^5_{t_4} \; O^5_{t_5}
\end{aligned}
$$

2. The most probable state sequences through the Single Look HMM's are determined for each of the above range profiles via the Viterbi algorithm. For this example, let the Single Look HMM's be 3 state left-right HMM's. Now suppose that the most probable state sequence through one the Single Look HMM's for all of the above range profiles is:

$$
\begin{aligned}
\boldsymbol{Q}^{*1} &= 1 \; 1 \; 1 \; 2 \; 3 \\
\boldsymbol{Q}^{*2} &= 1 \; 2 \; 2 \; 3 \; 3 \\
\boldsymbol{Q}^{*3} &= 1 \; 1 \; 1 \; 3 \; 3 \\
\boldsymbol{Q}^{*4} &= 1 \; 1 \; 2 \; 2 \; 3 \\
\boldsymbol{Q}^{*5} &= 1 \; 2 \; 3 \; 3 \; 3
\end{aligned}
$$

3. The state sequences are arranged left to right and top to bottom as follows:

$$
\begin{aligned}
O^{S_1} &= O^1_{t_1} \; O^1_{t_2} \; O^1_{t_3} \; O^2_{t_1} \; O^3_{t_1} \; O^3_{t_2} \; O^3_{t_3} \; O^4_{t_1} \; O^4_{t_2} \; O^5_{t_1} \\
O^{S_2} &= O^1_{t_4} \; O^2_{t_2} \; O^2_{t_3} \; O^4_{t_3} \; O^4_{t_4} \; O^5_{t_2} \\
O^{S_3} &= O^1_{t_5} \; O^2_{t_4} \; O^2_{t_5} \; O^3_{t_4} \; O^3_{t_5} \; O^4_{t_5} \; O^5_{t_3} \; O^5_{t_4} \; O^5_{t_5}
\end{aligned}
$$

From the above example, it is clear that the number of scattering centers that correspond to a given state will not necessarily be the same for all range profile sequences. More importantly, the Single Look HMM's for each class will generally segment the range profiles to different lengths for each state. Because identification is based the probability of the observation sequence given the model, shorter sequences have a clear advantage. Therefore, each state must be assigned a standard length for reliable classification. Thus, all state observation sequences that are longer than the standard length for that state are truncated to the preassigned length and state observation sequences which are shorter than the preassigned length are padded at the end by a null symbol.

The final classification decision is based on the comparison of the sums of the log probabilities of the 5 state HMM's for each class and it occurs after a predetermined number of range profiles have been received and processed.

3.5.6 Classification Based on Uniform State Transitions of Multiple Range Profiles This classifier is similar to the classifier described above except, the segmentation of the range profiles do not depend on the best state sequence of the Single Look HMM's. Instead, the range profiles are divided evenly. For example, if each range profile contains 10 scattering centers, the first two scattering centers will correspond to the state 1 HMM in Figure 3.4, the third and fourth scattering centers correspond to the state 2 HMM, and so on. The null symbol is not required for this classifier because each sequence will have a deterministic number of observations. Again the classification decision is based on the comparison of the sums of the log probabilities of the 5 state HMM's for each class and it occurs after a predetermined number of range profiles have been received and processed.

3.6 Summary

In this chapter the RTI algorithm has been described in detail. The front-end signal processing and feature extraction processing is common to all of the classification algorithms to be implemented

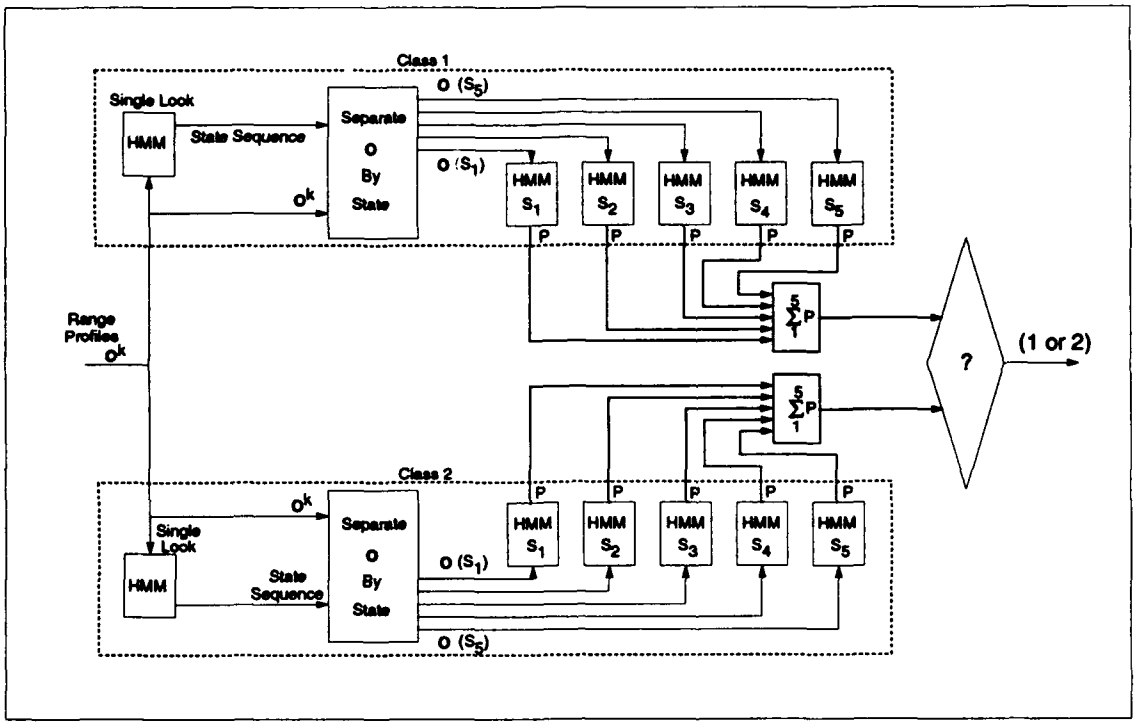


Figure 3.4. Multiple State Sequential Range Profile HMM Classifier

for this thesis. The vector quantization procedure reduces scattering centers to a single code book entry or the entire range profile to a single code book entry depending on the type of classifier implemented. In all, 5 classifiers will be implemented for this thesis. The experimental results pertaining to all phases of the RTI process are given in Chapter IV.

IV. EXPERIMENTAL RESULTS

4.1 Introduction

The theoretical basis for the RTI algorithms developed for this thesis is given in Chapter II, and Chapter III contains a general description these algorithms. The purpose of this chapter is to document the experimental results pertaining to all phases of the RTI process. Accordingly, this chapter is arranged as follows:

- Front-end signal processing and feature extraction.
- Vector quantization.
- HMM training verification.
- Classification based on a single range profile.
- Classification based on multiple vector quantized range profiles.
- Classification based on multiple state separated range profiles.
- Classification based on all of the above three classification schemes.

Conclusions and recommendations based on these results are discussed in Chapter V.

4.2 Front-End Signal Processing and Feature Extraction

The purpose of this section is to report the experimental results which verify the front-end signal processing and feature extraction processing which are common to all of the classification algorithms implemented for this thesis. The front-end signal processing consists of transforming from linear to circular polarization, filtering by a linear FM pulse compression transfer function, corrupting by AWGN, and then filtering by a filter matched to the linear FM pulse compression filter. The Prony model parameter estimation is the heart of the feature extraction processing.

4.2.1 Processing of a Known Idealized Range Profile The first part of this experiment is conducted with a known idealized range profile of six point scatterers. The point scatterers are sampled

at 8 times the bandpass Nyguist rate to improve the granularity of the ensuing plots. The equation to generate the point scatterers is expressed in the discrete frequency domain as follows:

$$\begin{aligned} X_{xy}[k] &= \left(e^{j\omega_k \frac{N}{8}} + e^{j\omega_k \frac{N}{4}} + e^{j\omega_k \left(\frac{N}{4}+16\right)} + e^{j\omega_k \frac{N}{2}} + e^{j\omega_k \left(\frac{N}{2}+32\right)} + e^{j\omega \left(N-\frac{N}{16}\right)}\right) \\ &\quad \cdot \text{rect}\left[\frac{k - \left(f_c T_s N - \frac{B}{2\Delta f}\right)}{\frac{B}{\Delta f}}\right] \quad k = 0, 1, 2, \ldots, N-1 \end{aligned} \tag{4.1}$$

$$\begin{aligned} B &\equiv 1.5 \text{ GHz: bandwidth} \\ f_s = \frac{1}{T_s} &\equiv 8 \cdot B = 12 \text{ GHz: sampling rate} \\ \Delta f &\equiv \frac{c}{2R_E} = 5.859 \text{ MHz: frequency step size, } R_E = 25.6 \text{ m} \\ N &\equiv \frac{f_s}{\Delta f} = 2048\text{: number of discrete frequency samples} \\ \omega_k &\equiv 2\pi f_s T_s \frac{k}{N} = 2\pi \frac{k}{N}\text{: normalized discrete frequency} \\ xy &\equiv vv, vh, hh, \text{or} hv\text{: polarization} \end{aligned} \tag{4.2}$$

The IFFT of $X_{xy}[k]$ has 6 point targets at 3.2 m, 6.4 m, 6.6 m, 12.8 m, 13.2 m, and 24 m. A plot for a single polarization of the IFFT of $X_{xy}[k]$ is shown in Figure 4.1. In this plot, the range extent is 25.6 meters and the number of sample points is 2048, which corresponds to 8 samples per 0.1 meters in the discrete range domain. The RTI system block diagram shown in Figure 3.1 gives the general signal flow. Thus, the 4 linear polarized signals are transformed to 2 left circularly polarized signals by equation (2.11) to yield

$$X_{hl}[k] = \frac{1}{\sqrt{2}}\left(X_{hh}[k] + \mathrm{j}X_{hv}[k]\right)$$

$$X_{vl}[k] = \frac{1}{\sqrt{2}}\left(X_{vv}[k] + \mathrm{j}X_{hv}[k]\right).$$

At this point, $X_{hl}[k]$ and $X_{vl}[k]$ are multiplied by an over sampled linear FM transfer function, $F[k]$. $F[k]$ is the FFT of equation (3.3) where $T_s = 8 \cdot B$ and $n = 0, 1, 2, \ldots, 2047$. Without adding any noise, the 2 signals are then matched filtered by $F^*[k]$. Noise is not added to these signals so that the

effects of the filtering and feature extraction can be clearly seen. To stay consistent with the notation, the signals out of the matched filter are $S_{vl}[k]$ and $S_{hl}[k]$.

Figure 4.2 shows the point scatterer at 3.2 m for a single polarization before and after filtering by the linear FM pulse compression transfer function and matched filter. The distance between the peak and the first null of the unfiltered signal is 0.1 m, which is the expected range resolution for a bandwidth of 1.5 GHz. After filtering by the 750 MHz linear FM transfer function and matched filter, the shape of the pulse is about the same but the range resolution is approximately 0.2 m. Figure 4.3 shows that 2 point scatterers separated by 0.2 m can be separated if the bandwidth is 1.5 GHz. However, after the matched filter the two scatterers appear as 1 scatterer. Referring to Figure 3.2, the actual bandwidth is slightly less than 750 MHz and hence the actual range resolution is just over 0.2 m. In Figure 4.4 the point scatterers at 12.8 m and 13.2 m are clearly separated before and after filtering. Although all of the point scatterers are generated with a magnitude of 1, interaction between the scatterers, especially after the matched filter, causes the relative amplitudes to vary.

$S_{vl}[k]$ and $S_{hl}[k]$ are identical in magnitude and phase, therefore the six point scatterers are circularly polarized. Before the Prony model parameter estimation procedure, $S_{vl}[k]$ and $S_{hl}[k]$ are truncated to the 116 frequency samples well within the bandwidth of $F[k]$. The resulting Prony model parameter estimation output is shown in Figure 4.5. All of the point scatterers have a circular polarization ellipse. Also, the 2 point scatterers at 3.2 and 3.4 meters are clearly separated even though the parameter estimation procedure inputs had bandwidths considerably less than 750 MHz.

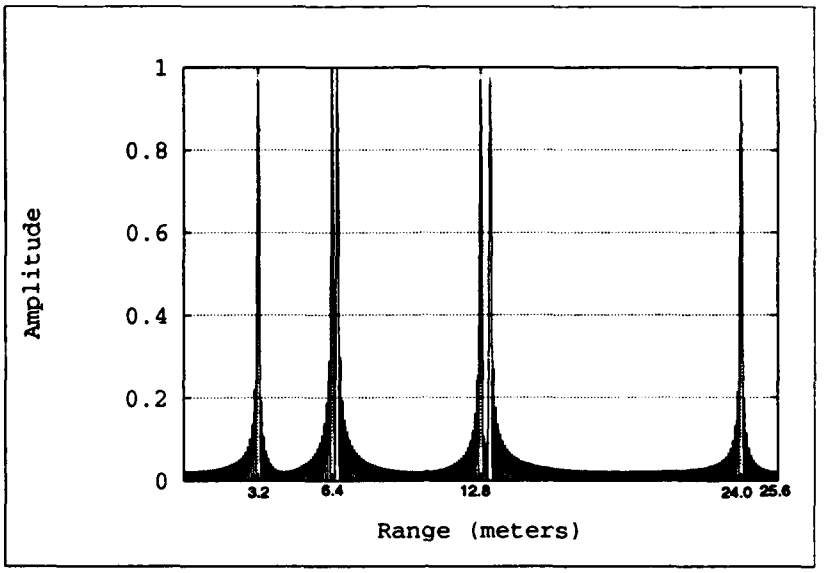


Figure 4.1. Six Point Scatterers (Uniform Bandwidth of 1.5 GHz)

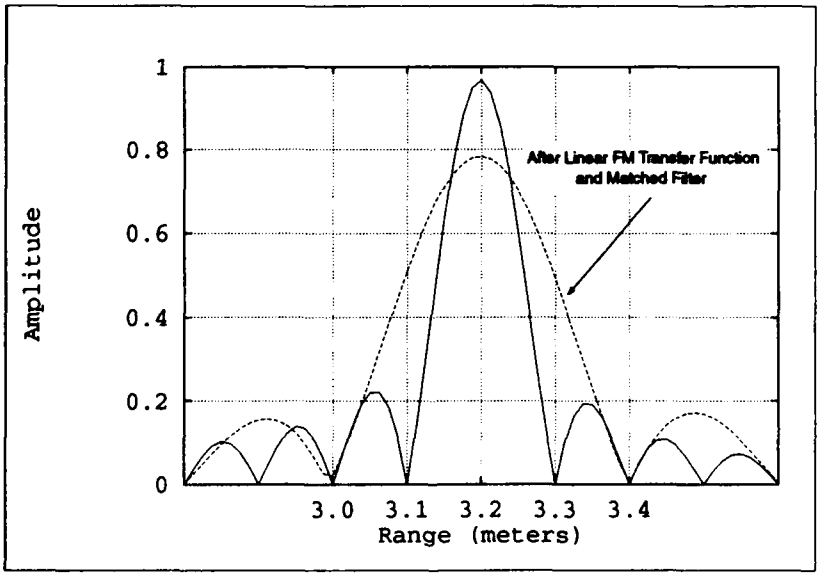


Figure 4.2. Point Scatterer at 3.2 meters, Before and After Linear FM Pulse Compression

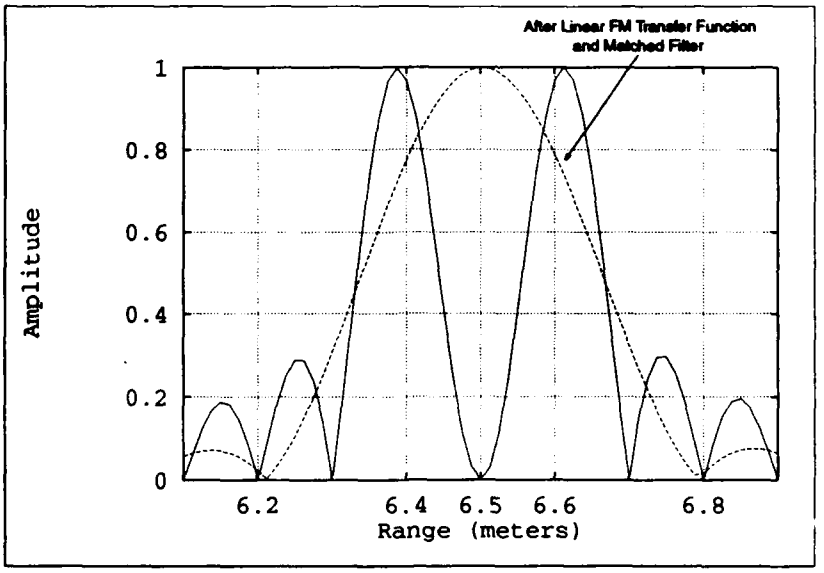


Figure 4.3. Point Scatterers at 6.4 and 6.6 meters, Before and After Linear FM Pulse Compression

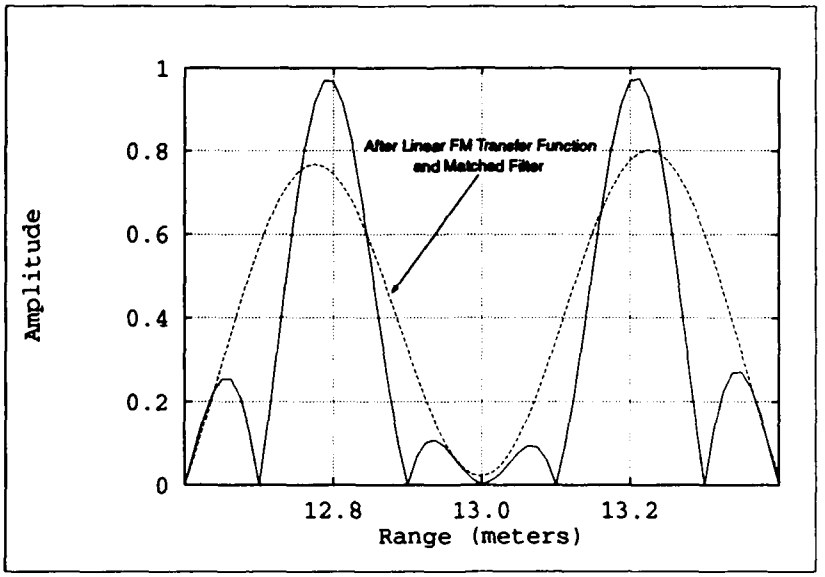


Figure 4.4. Point Scatterers at 12.8 and 13.2 meters, Before and After Linear FM Pulse Compression

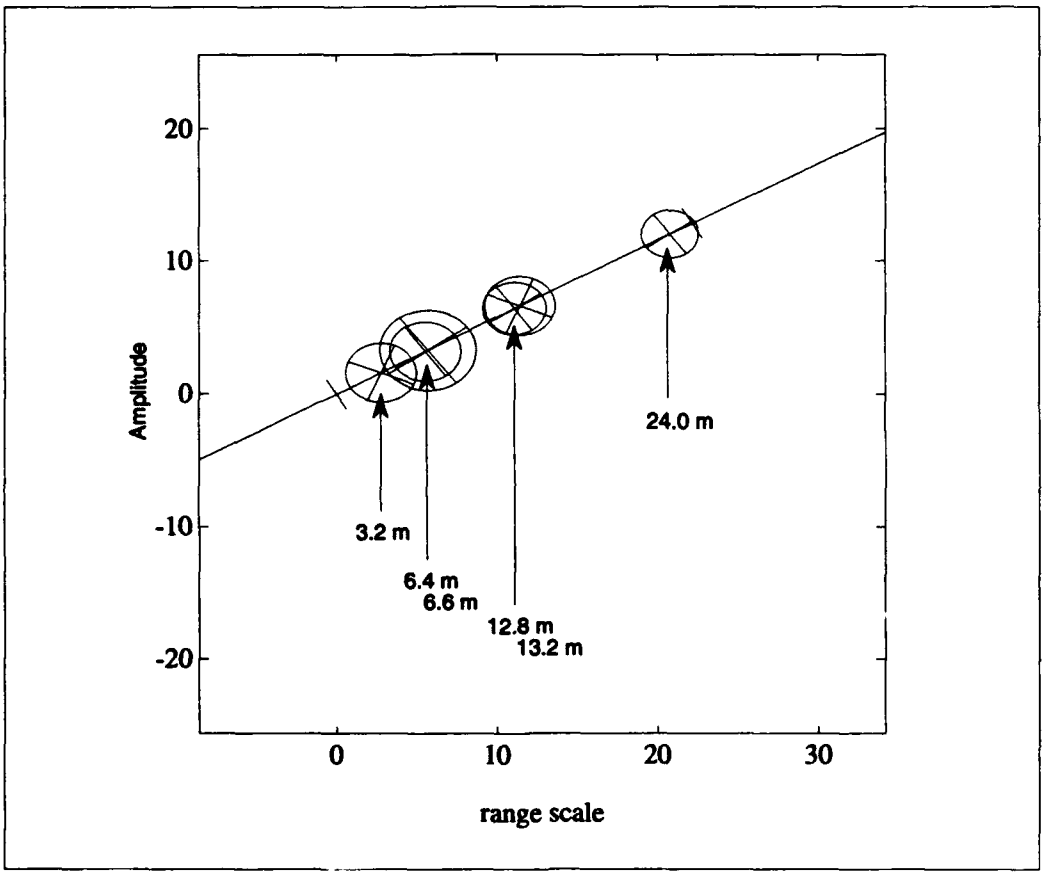


Figure 4.5. Six Circularly Polarized Point Scatterers Represented by the Prony Model

4.2.2 Processing Typical Xpatch Range Profiles The second part of this experiment is conducted using typical Xpatch range profiles. In this case, the front-end signal processing and feature extraction are performed as described in Sections 3.2 and 4.2 with a SNR of 20 dB and $f_s = 1.5$ GHz. The SNR is expressed in terms of peak signal power to average noise power. The noise corruption process is performed as follows:

1. The horizontal and vertical range profiles are transformed to the time domain using an IFFT routine and are normalized so that the largest scatterer has an amplitude of 1.
2. Independent Gaussian noise is added to each range bin. The SNR refers to the level of the noise compared to unit amplitude.
3. The range profiles are transformed back to the frequency domain.

Figure 4.6 shows typical range profiles after the matched filter from each or the two classes used for this thesis. Although the change in aspect angle is approximately 1°, the range profiles change dramatically. The scattering center at 12.8 m which is relatively large in Figure 4.6 A but it almost disappears in Figure 4.6 B. It is also important to note that the relative distance between the first, second and third scattering centers of the range profiles in Figures 4.6 B and D are almost identical. For this reason, these two targets are difficult to separate by classification algorithms which identify targets by matching peaks in the range profile to a templates. The GD algorithm described in Section 1.3.3 is as example of this type of algorithm. The SNR out of the matched filter is approximately 10 dB above the SNR ratio prior to the matched filter. All of the SNR's reported for the remainder of the experiments described in this chapter are prior to the matched filter.

Figures 4.7 A-E summarize the results of this experiment. The IFFT's horizontal and vertical range profiles out of the matched filter are shown in Figures 4.7 A and B respectively. The output of the feature extraction process is given graphically in a Figure 4.7 C and in tabular form in Figure 4.7 D. Figure 4.7 E defines the parameters of the polarization ellipses in Figure 4.7 C.

The scatterer at 12.3 meters has the highest energy despite the fact that it is not one of the larger scatterers in Figures 4.7 A and B. The reason for this is that the energy calculation, as given by equation (2.14), is greatly effected by $|p_{t_n}|$ because of the $|p_{t_n}|^{2k}$ factor. Therefore, scatterers with relatively large $|p_{t_n}|$ coefficients will have more energy. The scatterer at 12.3 meters has the

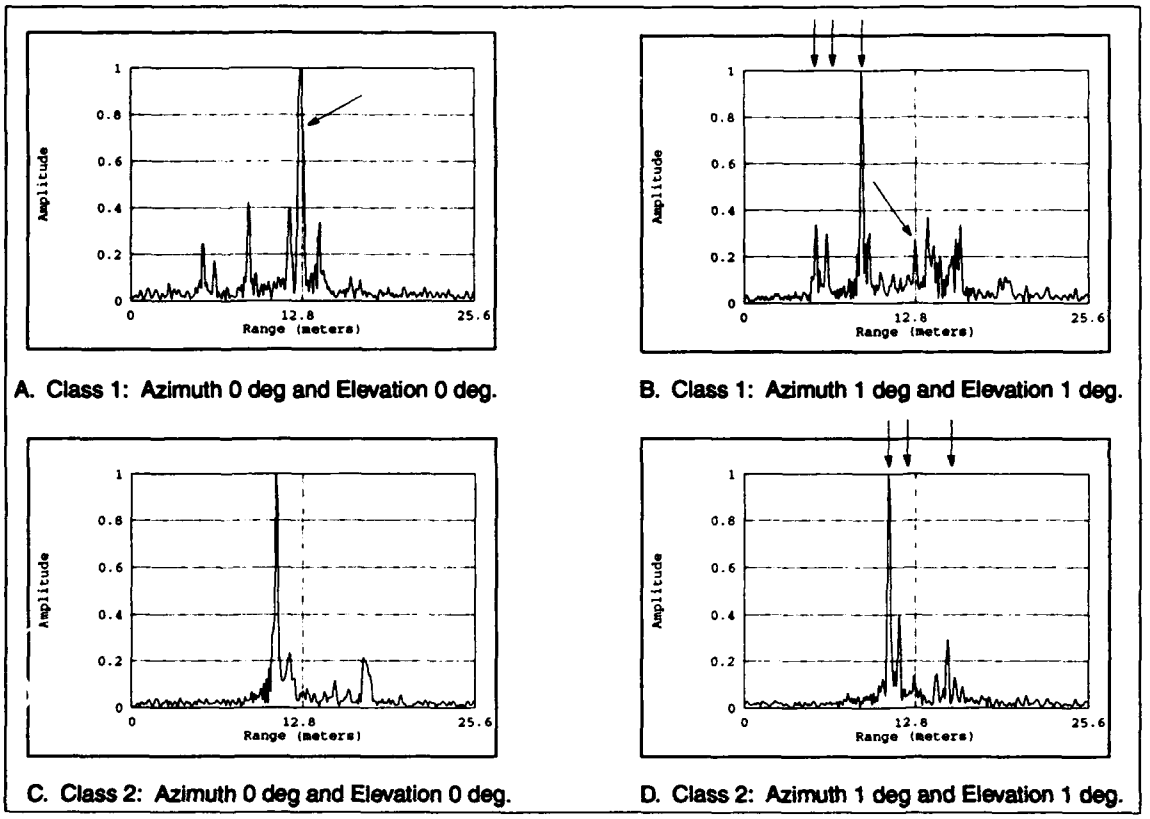
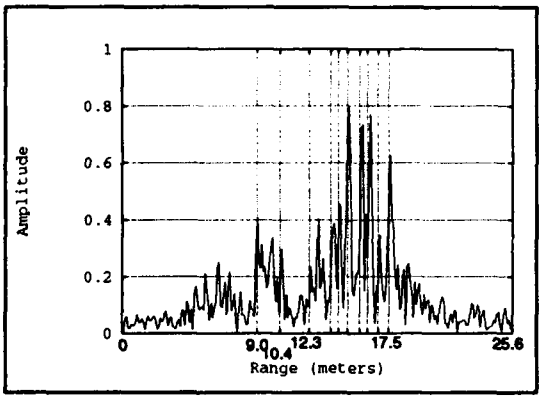


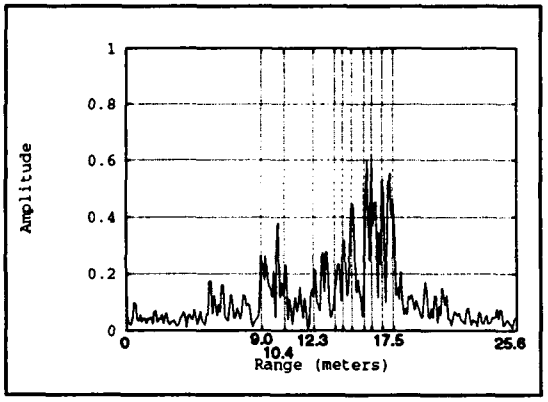
Figure 4.6. Typical Range Profiles from Class 1 and Class 2 After the Matched Filter

greatest energy because it has the largest $|p_{t_n}|$ coefficient ($|p_{t_n}| = 1.044$) as shown in Figure 4.7 D. This phenomenon is further illustrated by the 6 scatterers between 13.7 and 16.8 meters. All of these scatterers have relatively high amplitudes in the IFFT range profiles, but they have small $|p_{t_n}|$ coefficients and extremely low energy – low enough so that their polarization ellipses are too small to be seen in Figure 4.7 C.

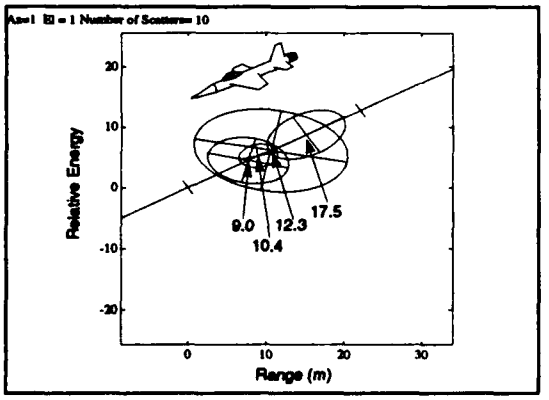
Its important to note that the amplitudes of the scattering centers of the horizontal IFFT range profile in Figure 4.7 A are generally larger than the amplitudes of the scattering centers of the vertical IFFT range profile in Figure 4.7 B. Accordingly, the orientations of the major axes (OA) of the polarization ellipses in Figure 4.7 C are generally tilted toward the horizontal h axis.



A. *hl* Range Profile (IFFT of $S_{hl}[k]$)



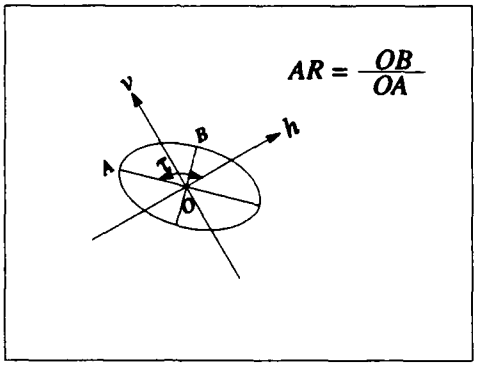
B. *vl* Range Profile (IFFT of $S_{vl}[k]$)



C. Prony Model Range Profile

|ρ|	*AR*	τ (deg)	*E* (norm)	*r* (m)
1.036	0.705	137	0.527	9.0
1.040	0.762	165	0.251	10.4
1.044	0.668	139	1.000	12.3
1.011	0.506	148	2.59e-4	13.7
1.020	0.638	119	7.62e-3	14.2
1.003	0.681	142	2.34e-4	14.8
1.009	0.821	132	7.48e-4	15.6
1.003	0.114	128	8.92e-5	16.1
1.012	0.396	122	2.53e-4	16.8
1.035	0.624	0	0.546	17.5

D. Feature Extraction Output



E. Polarization Ellipse

Figure 4.7. Front-End Processing and Feature Extraction for an Actual Range Profile

4.2.3 Range Profile Computing Time Both the front-end processing and feature extraction processing algorithms are implemented in MATLAB and each Xpatch signature takes 10 to 15 seconds to process on a Sun Sparcstation 2 workstation. MATLAB is inherently slow because it interprets the program a line at a time instead of compiling the program into machine language. The majority of the computing time, however, is spent doing the SVD, which MATLAB has implemented in machine language. As will be shown in the following section, the amount of processing time per range profile has far reaching implications.

4.3 Vector Quantization

The next step in the RTI process is vector quantization. As discussed in (22) and shown in the next section, there is a definite trade-off between the number of clustering centers or number of code book symbols and quantization error. That is, the B matrix of the HMM must have as many columns as code book symbols. Increasing the size of the B matrix increases the number of parameters that need to be estimated during the HMM training procedure which in turn increases the size required for the training set. Thus, the purpose of this experiment is to determine the quantization error versus M, number of code book symbols.

Figure 4.8 shows the average quantization error versus M (on a log scale) where each scattering center of the range profile is vector quantized. The clustering centers are found using the K-means clustering option in LNKnet. In all, 280,000 normalized scattering centers (140,000 from each class) from 28,000 range profiles are used for this experiment. As Figure 4.8 clearly shows, the average quantization error decreases linearly as M increases between 16 and 64. For M greater than 64, however, the average quantization error decreases less between 128 and 1024.

The same experiment is conducted for the range profile vector quantization as shown in Figure 4.9. In this experiment a 50 dimensional feature vector is vector quantized. The relationship between the average quantization error and M is linear for M less than or equal to 1024. The information learned from this experiment and the HMM training verification discussed in the next section is used to determine the size of HMM's.

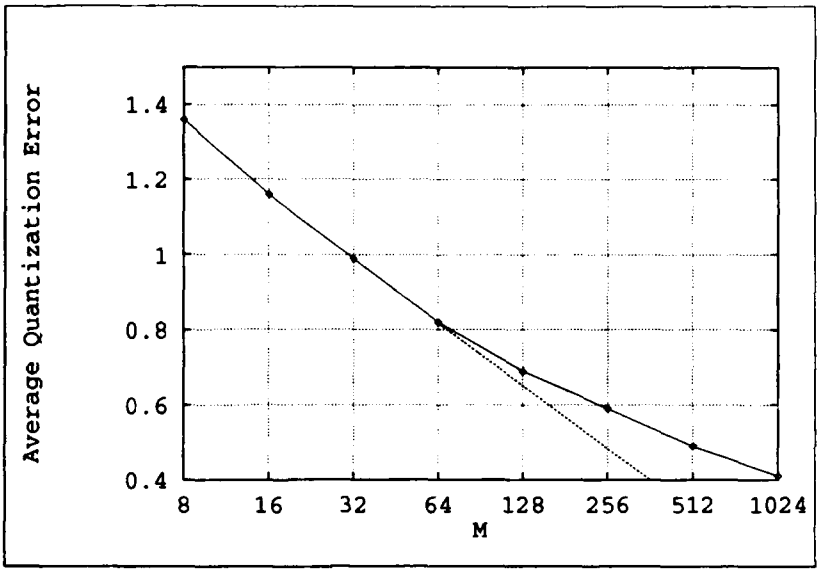


Figure 4.8. Scattering Center Vector Quantization Error

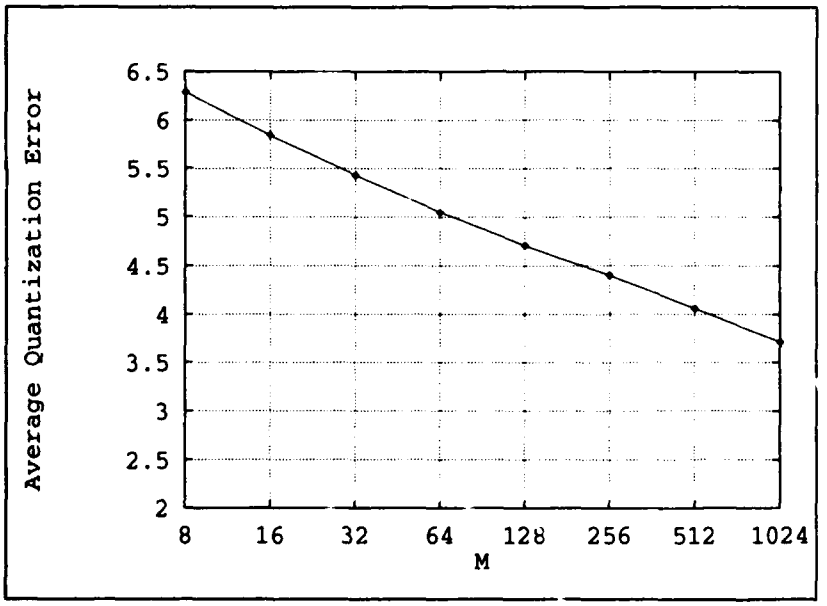


Figure 4.9. Range Profile Vector Quantization Error

4.4 HMM Synthesis

With respect to the HMM classification algorithms developed for this thesis, the most critical process is the solution to Problem 3 (ref Section 2.4.4) – the estimate of the parameters of the HMM's given the observations. Of the 3 HMM design problems stated in Section 2.4.4, Problem 3 is also the most difficult because, unlike the other 2 problems, it does not have an exact analytical solution. With this in mind, the purpose of the experiments in this section are two fold:

1. To gain insight as to how much training data is required to adequately train the HMM's.
2. To insure that the training procedure implemented provides reasonable estimates of the HMM parameters.

4.4.1 Training Data Requirements The amount of training data required depends on the number of model parameters (22). To make a rough estimate of the amount of training data required, this experiment is conducted as follows. First, two hypothetical left-right HMM's are created. Let the first model be labeled 1 and let the second model be 1a. These model parameters are

1. Model –1: ($N = 5$ and $M = 32$)

$$\boldsymbol{\pi}_1 = \begin{pmatrix} 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{pmatrix} \tag{4.3}$$

and

$$\boldsymbol{A}_1 = \begin{pmatrix} 0.3 & 0.4 & 0.2 & 0.1 & 0.0 \\ 0.0 & 0.4 & 0.3 & 0.2 & 0.1 \\ 0.0 & 0.0 & 0.6 & 0.3 & 0.1 \\ 0.0 & 0.0 & 0.0 & 0.4 & 0.6 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}. \tag{4.4}$$

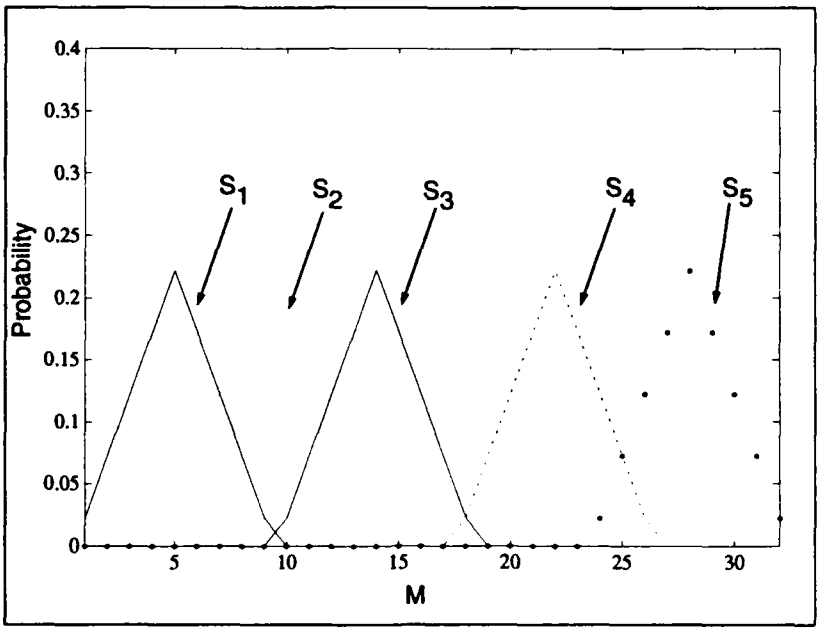


Figure 4.10. Model –1 B_1 Matrix Probabilities

2. Model – 1a : ($N = 5$ and $M = 128$)

$$\pi_{1a} = \begin{pmatrix} 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{pmatrix} \tag{4.5}$$

and

$$A_{1a} = \begin{pmatrix} 0.3 & 0.4 & 0.2 & 0.1 & 0.0 \\ 0.0 & 0.4 & 0.3 & 0.2 & 0.1 \\ 0.0 & 0.0 & 0.6 & 0.3 & 0.1 \\ 0.0 & 0.0 & 0.0 & 0.4 & 0.6 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}. \tag{4.6}$$

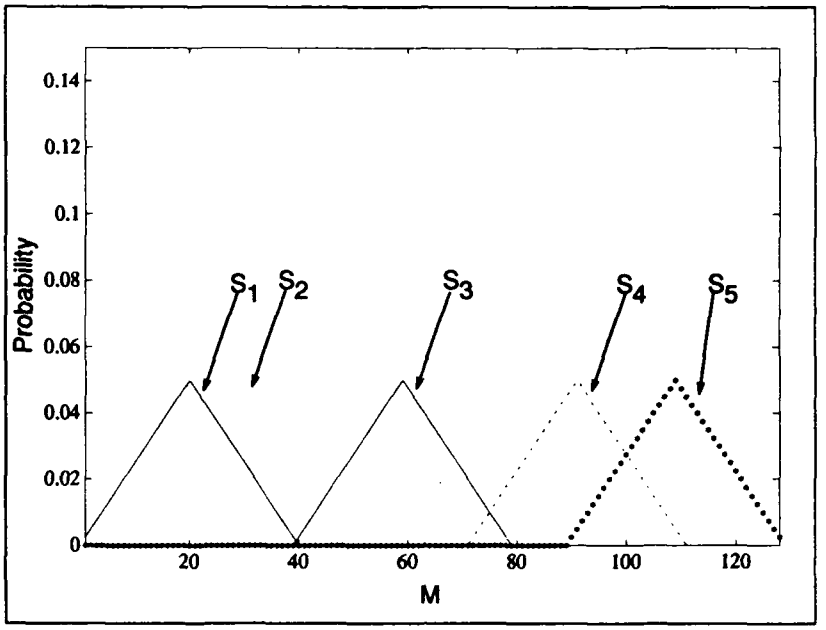


Figure 4.11. Model –1a B_{1a} Matrix Probabilities

Next, Models 1 and 1a are used to generate observation sequences using the procedure given in Section 2.4.3. Each observation sequence has of 10 observations. Figure 4.11 shows a histogram of the generated observations that correspond to each state normalized by the total observations that occur in that state. Ideally, the normalized histogram of the generated observations should be about the same as the representation of B_1 matrix shown in Figure 4.10. For Model 1, it appears that at stochastic properties of the model are not well represented for observation sequences of 1000 observations or less. That is to say, if less than 1000 observation sequences are used to synthesize a model using the Baum-Welch training procedure the resulting model would show little resemblance to the model which generated the observation sequences. As can be seen by Figure 4.13, the number observation sequences required to adequately represent the model also increases. From this experiment, one could conclude that 1000 to 5000 observation sequences would be required to train Model 1 and 10000 to 20000 observation sequences would be required to train Model 1a. Although not shown here, the histograms of the A matrices for both models displayed similar properties.

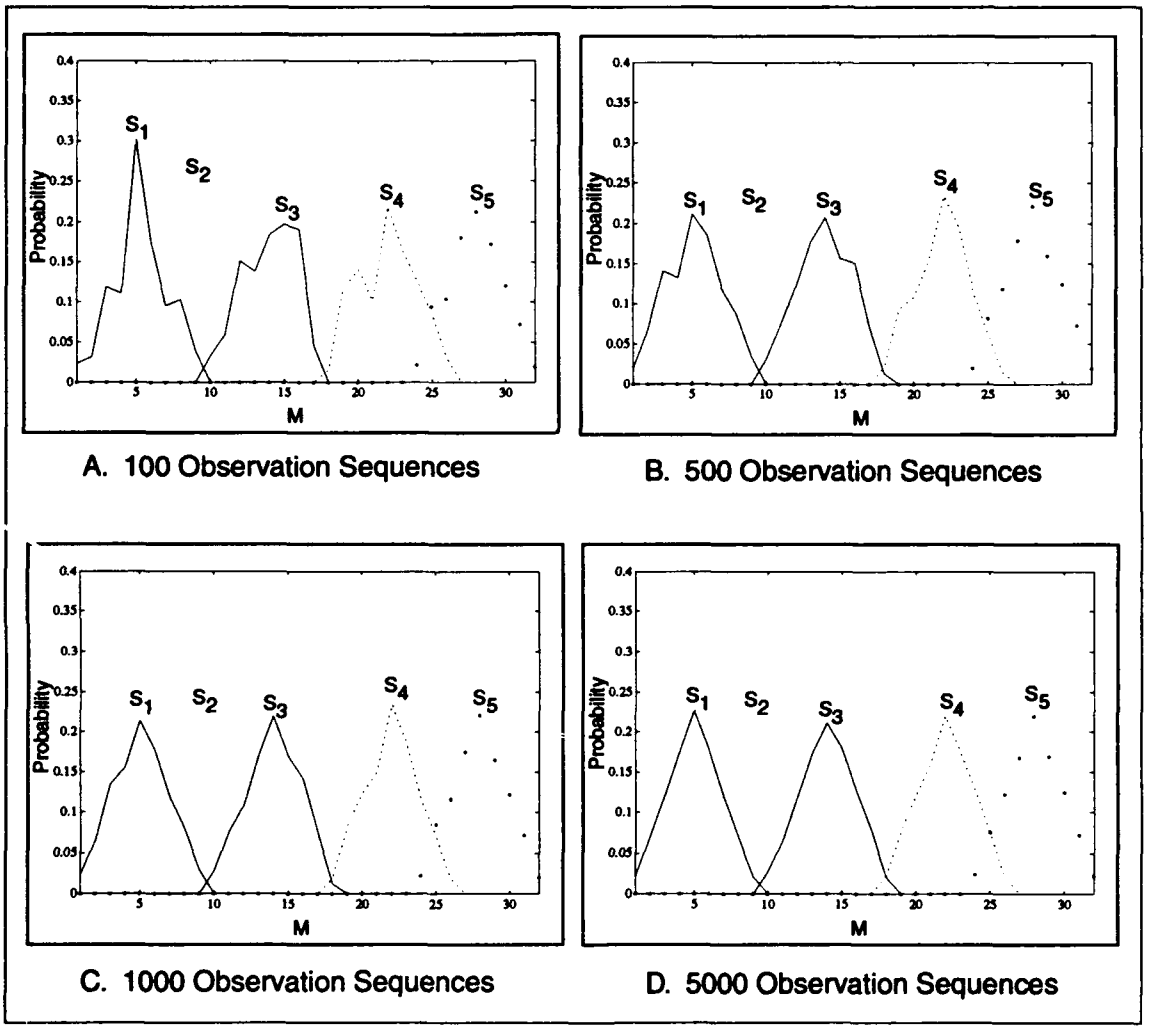


Figure 4.12. Model -1 Normalized Observation Histograms

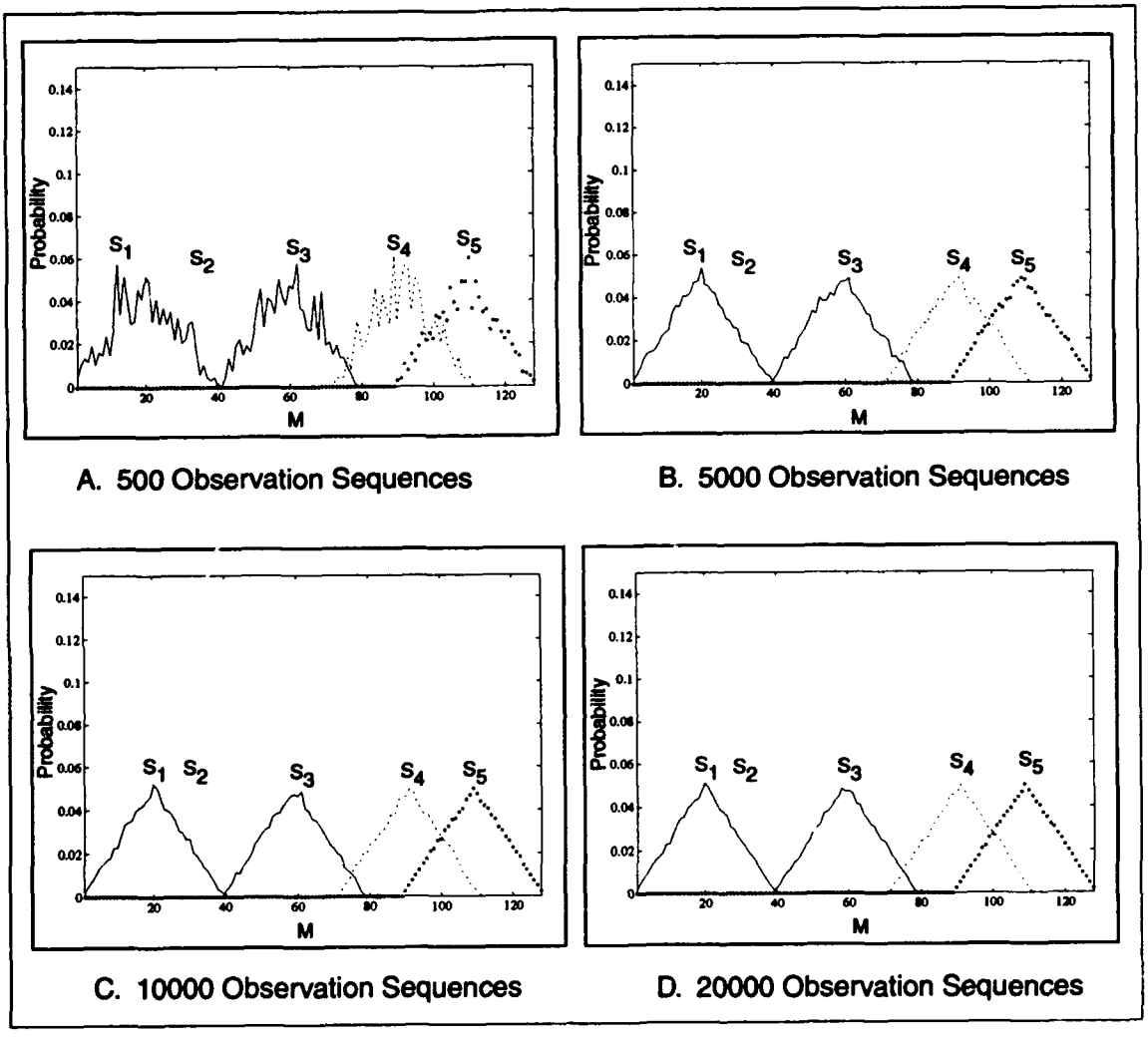


Figure 4.13. Model -1a Normalized Observation Histograms

4.4.2 HMM Training The objective of the second part of this experiment is to insure that the training procedure, which will be described below, yields reasonable estimates of the model parameters. To do this, 2 additional left-right models are created, where $N = 5$ and $M = 32$. Let these models be labeled Model –2 and Model –3. Model – 1 is the same as above. The B matrices of the 3 models are shown in Figures 4.14 – 4.16 A. The A matrix for models 2 and 3 are

$$A_2 = \begin{pmatrix} 0.3 & 0.4 & 0.2 & 0.1 & 0.0 \\ 0.0 & 0.4 & 0.3 & 0.2 & 0.1 \\ 0.0 & 0.0 & 0.6 & 0.3 & 0.1 \\ 0.0 & 0.0 & 0.0 & 0.4 & 0.6 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix} \tag{4.7}$$

and

$$A_3 = \begin{pmatrix} 0.04 & 0.09 & 0.26 & 0.28 & 0.32 \\ 0.0 & 0.25 & 0.27 & 0.24 & 0.23 \\ 0.0 & 0.0 & 0.06 & 0.68 & 0.26 \\ 0.0 & 0.0 & 0.0 & 0.13 & 0.87 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}. \tag{4.8}$$

The 3 models are used to generate 10000 sequences (each with 10 observations) using the procedure given in Section 2.4.3. These observation sequences are then used to train 3 models using the training procedure developed for this thesis. This procedure is as follows:

1. Start with a uniformly distributed left-right model, λ. That is,

$$A = \begin{pmatrix} 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.0 & 0.25 & 0.25 & 0.25 & 0.25 \\ 0.0 & 0.0 & 0.33 & 0.33 & 0.34 \\ 0.0 & 0.0 & 0.0 & 0.5 & 0.5 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix} \tag{4.9}$$

and

$$b_i(m) = \frac{1}{32}, 1 \leq i \leq 5 \text{ and } 1 \leq m \leq 32. \tag{4.10}$$

2. Compute $P(O|\lambda)$
3. Estimate the parameters of $\overline{\lambda}$ using equations (3.6) and (3.7).
4. Compute $P(O|\overline{\lambda})$.
5. Replace λ with $\overline{\lambda}$. (Note: All B matrix coefficients less 10^{-10} are changed to 10^{-10}. Rabiner has shown that not letting the B coefficients approach 0 improves the classification rate (22).)
6. If $P(O|\overline{\lambda}) - P(O|\lambda) \leq 0.01$ terminate the procedure; otherwise, return to step 2 and repeat the procedure.

The above training procedure is used to train 3 models; 1 for each observation sequence. The B matrices after training are given in Figures 4.14 – 4.16 B and the original B matrices and the B matrices after training are overlayed in part C of Figures 4.14 – 4.16. The A matrices after training are

$$\overline{A}_1 = \begin{pmatrix} 0.296 & 0.402 & 0.201 & 0.101 & 0.0 \\ 0.0 & 0.409 & 0.296 & 0.199 & 0.096 \\ 0.0 & 0.0 & 0.604 & 0.297 & 0.099 \\ 0.0 & 0.0 & 0.0 & 0.410 & 0.590 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}, \tag{4.11}$$

$$\overline{A}_2 = \begin{pmatrix} 0.294 & 0.408 & 0.196 & 0.098 & 0.004 \\ 0.0 & 0.421 & 0.261 & 0.175 & 0.143 \\ 0.0 & 0.0 & 0.607 & 0.319 & 0.074 \\ 0.0 & 0.0 & 0.0 & 0.395 & 0.605 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix} \tag{4.12}$$

and

$$\overline{A}_3 = \begin{pmatrix} 0.039 & 0.088 & 0.261 & 0.288 & 0.324 \\ 0.0 & 0.274 & 0.258 & 0.235 & 0.232 \\ 0.0 & 0.0 & 0.061 & 0.726 & 0.213 \\ 0.0 & 0.0 & 0.0 & 0.151 & 0.849 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}. \quad (4.13)$$

Clearly, the training procedure is able to synthesize models that are almost identical to the original models used to generate the observation sequences used for training. The most significant result of this experiment is that the models are estimated with no – with the exception of knowing that the models are left-right models – *apriori* knowledge of the original models.

In Figures 4.14 – 4.16 D, $P(O|\overline{\lambda})$ is plot as a function of training epochs. Model – 1, which has the least amount of overlap between states, trains with about 1/4 the number of epochs as Models – 2 and – 3. Also, $P(O|\overline{\lambda})$ increases rapidly during the first few training epochs for all three models, but for Models – 2 and –3 the slope of $P(O|\overline{\lambda})$ levels off and then increases sharply before leveling off again. A possible reason for this is that the $P(O|\overline{\lambda})$ surface is more complex for Models – 2 and –3 than it is for Model – 1. Using equation (3.6) allows the training procedure to search the $P(O|\overline{\lambda})$ surface for a more probable model then would be found if the training procedure was allowed to stop at a the first local maximum.

The next logical step is to see if the synthesized models can recognize observation sequences that are generated from the corresponding original models. To do this, each of original models is used to generate 200 observation sequences with 10 observations each. Each observation sequence is then scored by each of the synthesized models and the model with the highest probability given the observation sequence is declared the winner. Two measures are used for scoring:

- $P(O|\lambda)$ as calculated with the scaled forward algorithm.
- $P(O|Q^*, \lambda)$ or the probability of the observation sequence given the best state sequence and the model. $P(O|Q^*, \lambda)$ was calculated using the Viterbi algorithm.

The correct classification using the $P(O|\lambda)$ metric is 97.3 % and the correct classification for the $P(O|Q^*,\lambda)$ metric is 96.5 %. Also, the confusion matrices for the both probability measures are given in Tables 4.1 and 4.2.

Table 4.1. Confusion Matrix Using $P(O|\lambda)$

Actual Model	Chosen Model		
	Model – 1	Model – 2	Model – 3
Model – 1	192	2	6
Model – 2	0	199	1
Model – 3	7	0	193

Table 4.2. Confusion Matrix Using $P(O|Q^*,\lambda)$

Actual Model	Chosen Model		
	Model – 1	Model – 2	Model – 3
Model – 1	192	0	8
Model – 2	3	196	1
Model – 3	9	0	191

The difference in the classification rates for both decision metrics is not statistically significant. However, it is interesting to note that the primary source of confusion is between Model –1 and Model–3 even though Model – 1 and Model –2 have identical state transition matrices. From this it is apparent that the B matrix parameters are more critical than the A matrix parameters.

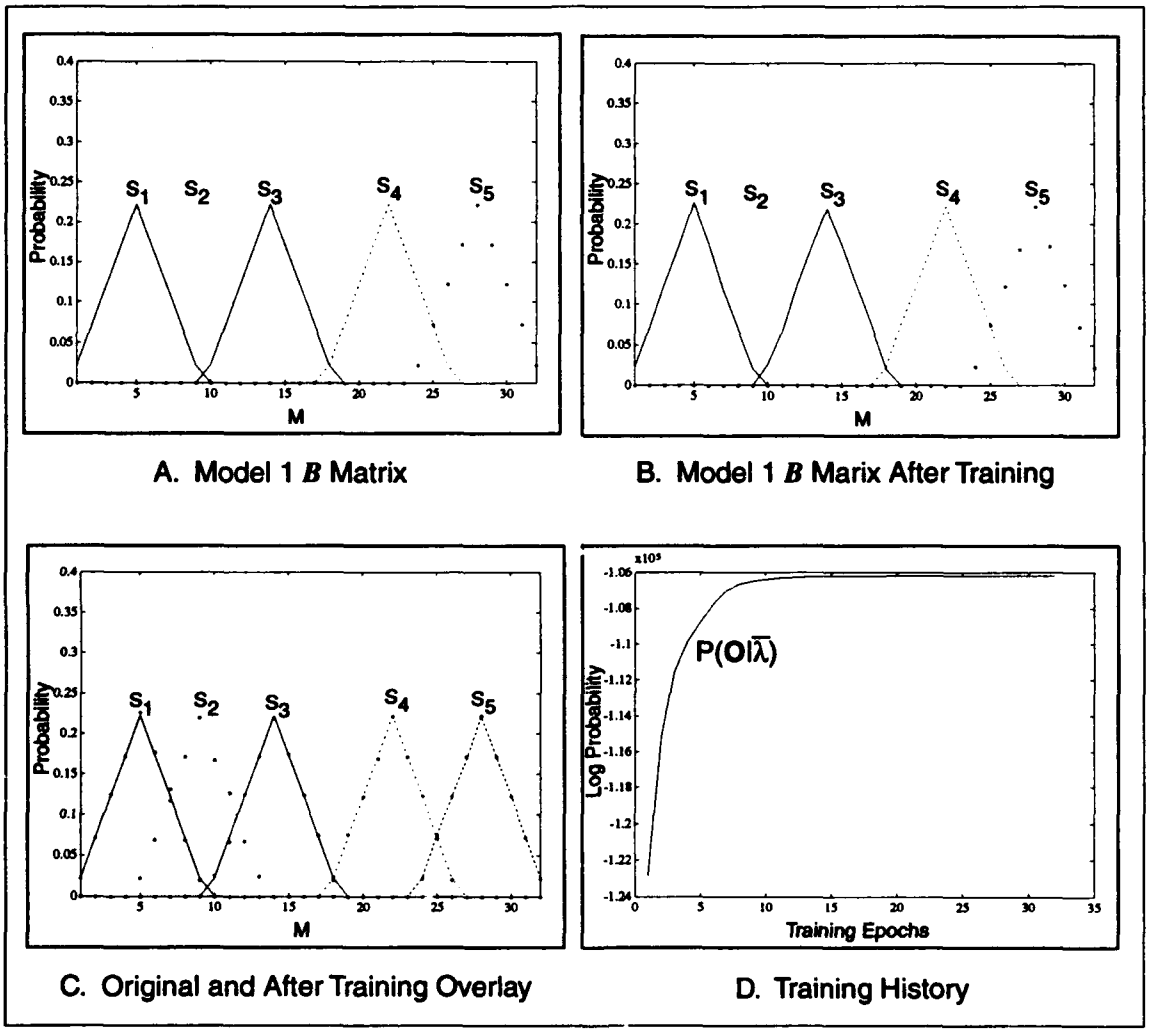


Figure 4.14. Model 1 Training

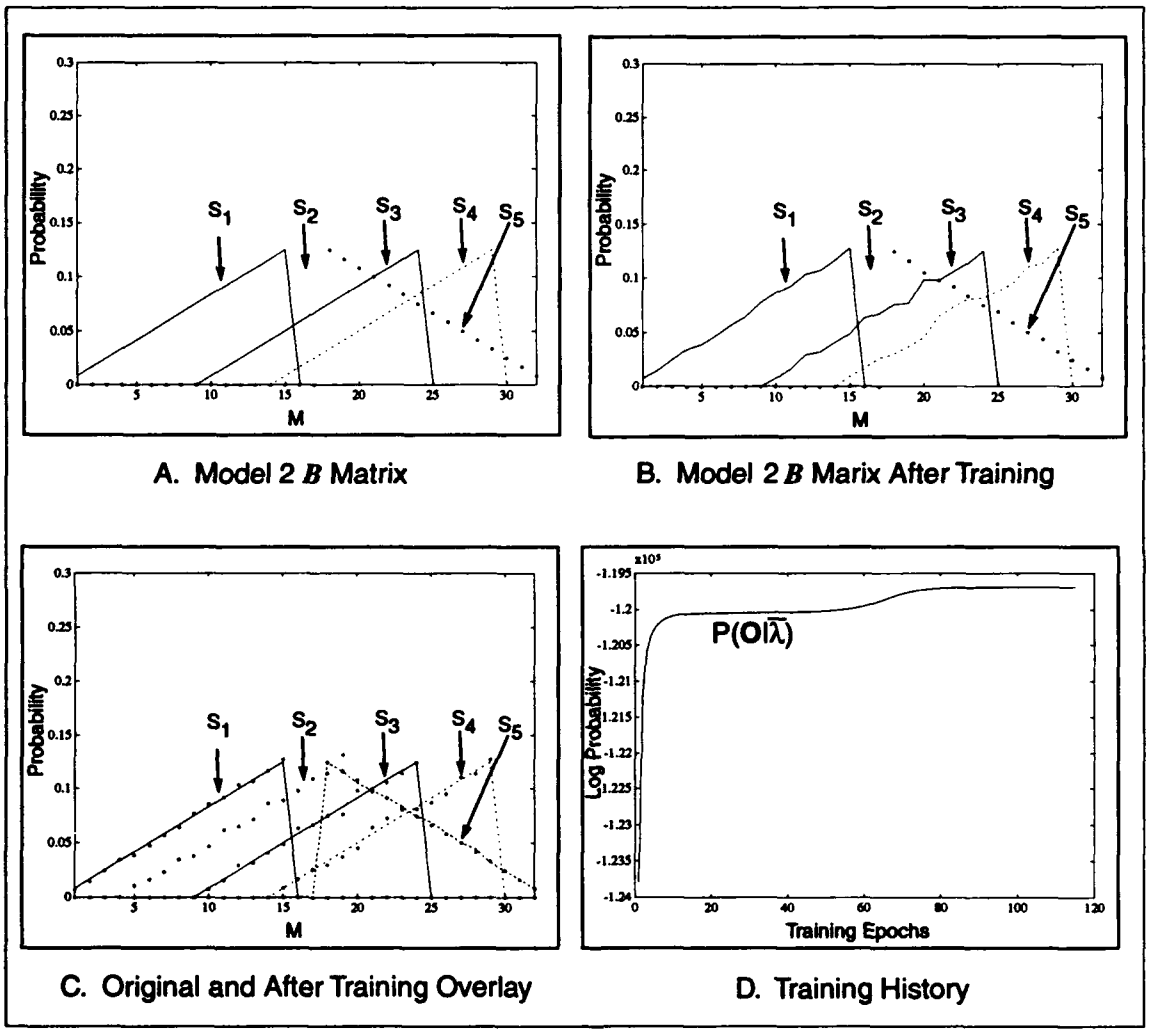


Figure 4.15. Model 2 Training

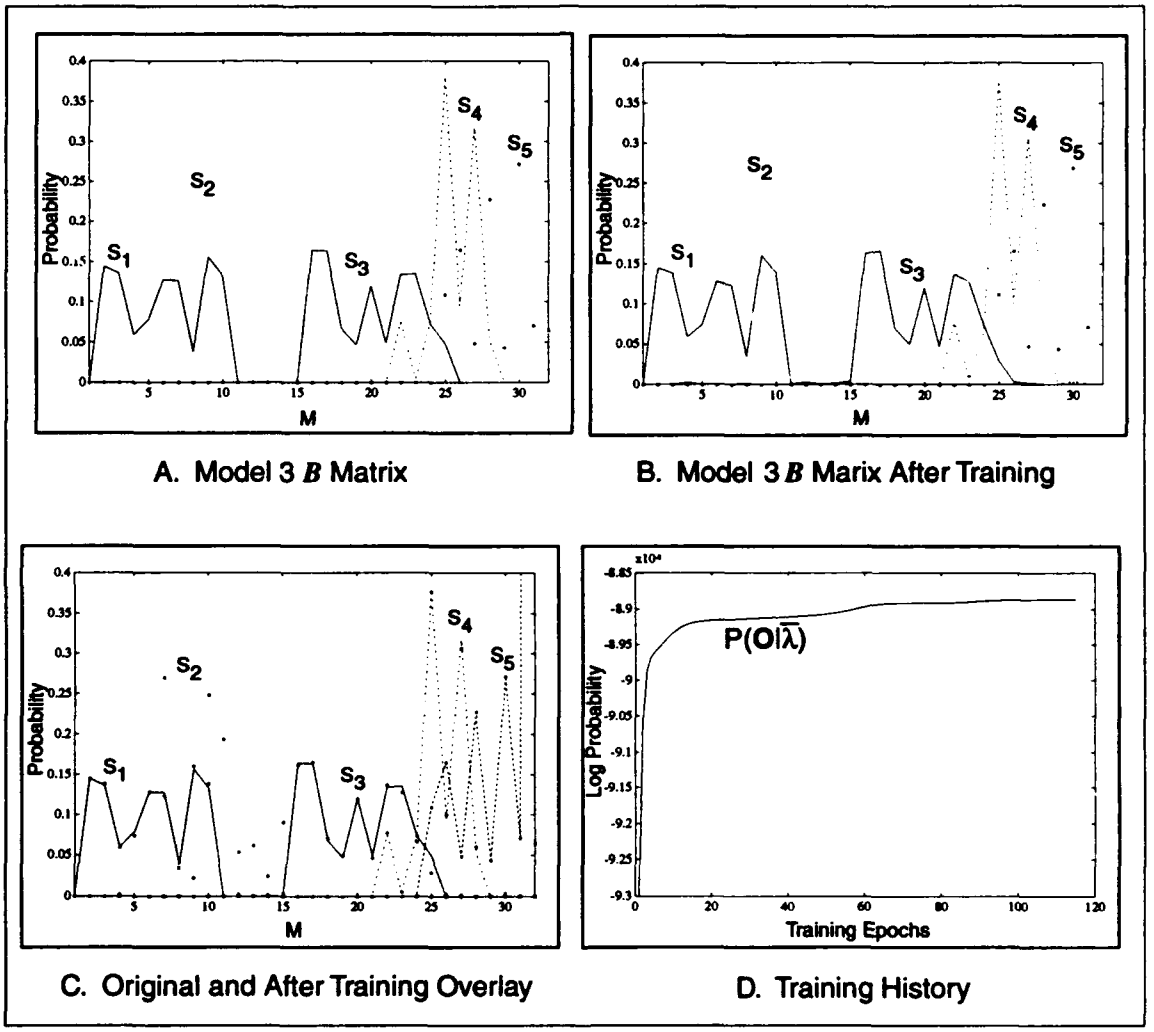


Figure 4.16. Model 3 Training

4.5 Classification Based on a Single Range Profile

For this experiment, each HMM of the single range profile classifier in Figure 3.3 is a left-right model with $N = 5$ and $M = 128$ (5 by 128). Rabiner's research showed that classification accuracies did not change appreciably for models with 5 or more states. For this reason, all of the HMM used for this thesis will have 5 states. On the other hand, the choice of M requires a trade-off between average vector quantization error and number of training sequences required.

Increasing M decreases the average vector quantization error, but the number of training vectors increase. Considering that it takes 10 to 15 seconds to process every range profile used for training, the number of range profiles required for training is a real issue. Referring to Figure 4.8, $M = 128$ is right at the beginning of the 'knee' of the curve and for $M \geq 256$ the average quantization error is nearly constant. Based solely on the average quantization error, $M = 256$ appears to be the best choice; but computing time constraints forces M to be no more than 128. Even for $M = 128$, the computing time to process the training data set is well over 100 hours.

The HMM for class 1 is trained using 14,000 vector quantized range profiles from vehicle 1. Each range profile consists of 10 scattering centers and have a SNR of 20 dB. In all, the training set consists of 140 noise independent range profiles for each of the 100 aspect angles within the aspect angle sector. The class 2 HMM is trained with the same type of data from vehicle 2. Both models are trained using the training procedure outlined in Section 4.4.2. The evaluation data set contains 600 profiles – 6 for each aspect angle – from each target at SNR's of 20 dB, 15 dB, 10 dB, and 5 dB, 0 dB, and - 5 dB.

As a comparative test, the same data set is processed by the K-nearest neighbor (KNN) classifier option in LNKnet. To do this, the range profiles from both the training and the evaluation data sets are concatenated so that each range profile is represented by a 50 dimensional feature vector. The LNKnet KNN classification algorithm is run with K equal to 1, 3, 5, and 7.

Figure 4.17 shows the classification results of both classifiers, for $M = 128$. As an indication of the accuracy of the error rate estimates, the classification confidence interval for a classification rate of 77 % is $\pm 1.7\%$ with a confidence level of 95 % is shown in the bottom right corner of Figure 4.17. Although the performance of the KNN classifier is better than the Single Look HMM at 20 dB SNR, the HMM classifier displays better performance for SNR's less than 20 dB. The performance of the

KNN classifier is not greatly effected by the value of K which is an indication that decision regions within the 50 dimensional feature space are relatively dense.

To test the effects of range alignment on the classification performance, the target is allowed to slide within the range window according to a Gaussian distribution with 0 mean and a variance of 2 m. The tracking system is assumed to be accurate to within this 2 m window (16). As Figure 4.18 shows, the classification accuracy drops about 10 %.

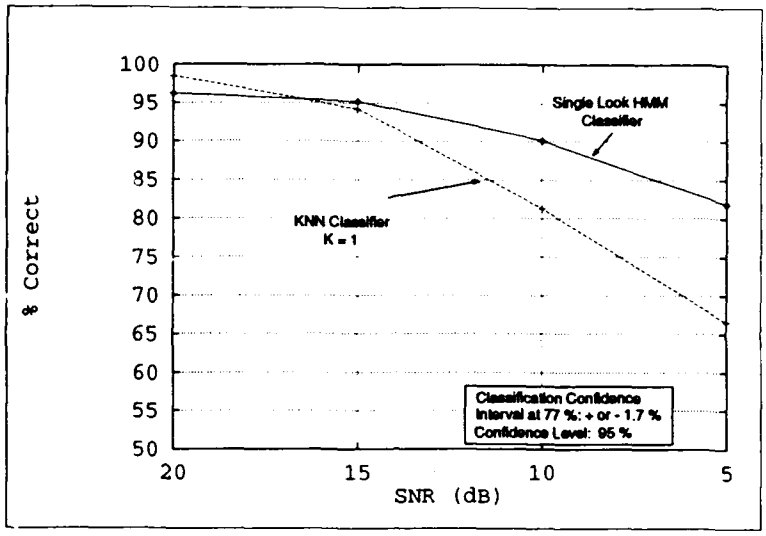


Figure 4.17. Classification Based on Single Range Profiles

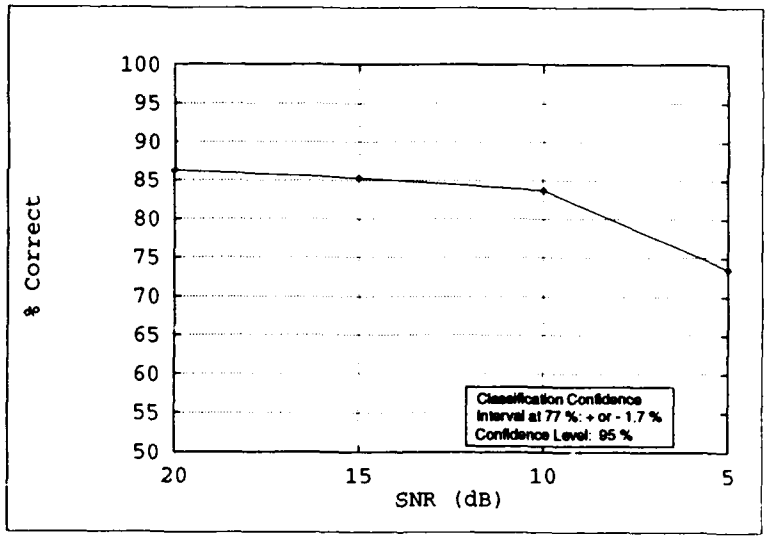


Figure 4.18. Classification Based on Single Range Profiles which are not Centered within the Range Window

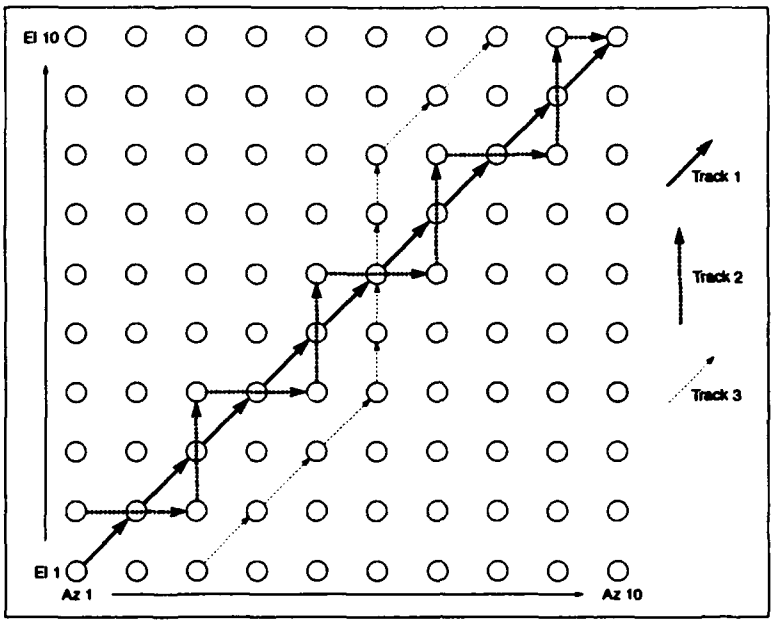


Figure 4.19. Aspect Angle Tracks 1 through 3

4.6 Classification Based on Sequences of Range Profiles

Three of the classification algorithms developed for this thesis are specifically designed recognize the target based on the inter range profile temporal relationship within a sequence of range profile. These algorithms are described in Sections 3.5.4, 3.5.5, and 3.5.6. As described in Section 3.5.3, the Single Look classifier makes its decision based on the the sum of the log probability outputs over several range profiles.

All of the range profile sequences used to train and test the classifiers have a fixed length of 10 range profiles. In all 7 tracks (sequences of range profiles through the aspect window) are used. These tracks are shown in Figures 4.19 and 4.20. Only tracks 1 and 4 are used for training, but all 7 tracks are used to test the performance of the classifiers. Specifically, the training set for each class consists of 3000 range profile sequences: 1500 from track 1 and 1500 from track 4. Each sequence contains 10 range profiles with SNR's of 20 dB. The evaluation data set has 375 range profile sequences from tracks 1 – 7 at SNR's of 20 dB, 15 dB, 10 dB, 5 dB, 0 dB, and -5 dB. With the exception of the Single Look classifier which is trained using the data set described in Section 4.5, these training and evaluation data sets are used in each of the following sequential range profile classifiers.

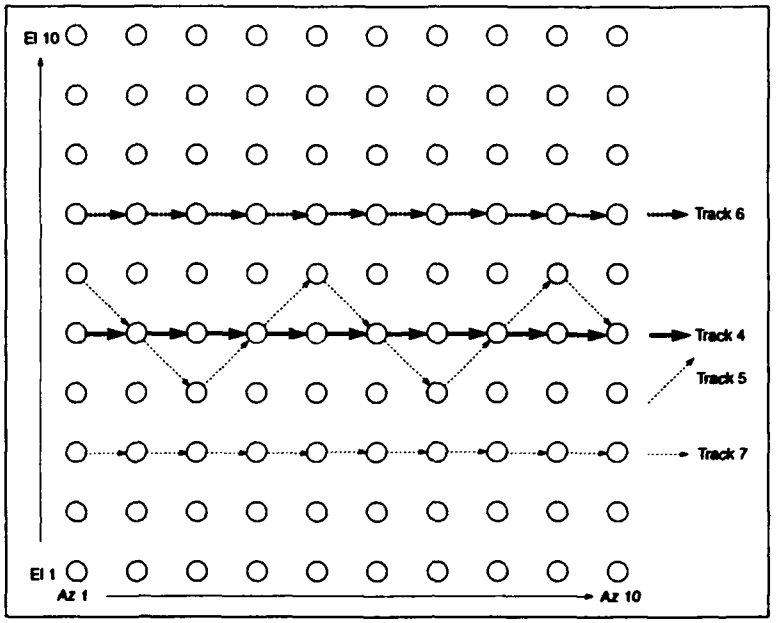


Figure 4.20. Aspect Angle Tracks 4 through 7

4.6.1 Performance of the Single Look Classifier after Multiple Range Profiles The performance of this classifier for tracks 1 and 4 and tracks 2, 3, 5, 6, and 7 is shown in Figures 4.21 and 4.22. This classifier is track independent because it is trained on independent range profiles, but at 0 dB SNR there is a significant difference in classification rates between tracks. Thus, there are tracks in which these two targets are harder to separate. The two targets appear to be closest within the feature space for track 6, which has the lowest classification rate as highlighted in Figure 4.22.

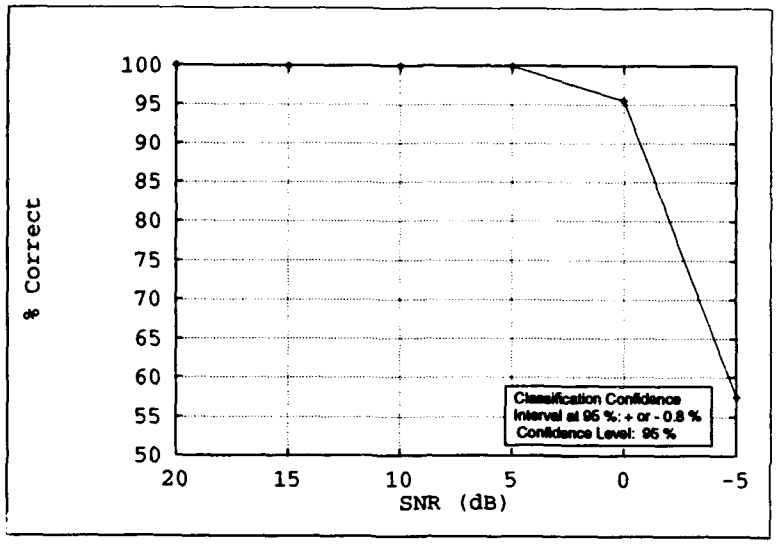


Figure 4.21. Single Look Multiple Range Profile Classification Rate (Tracks 1 and 4)

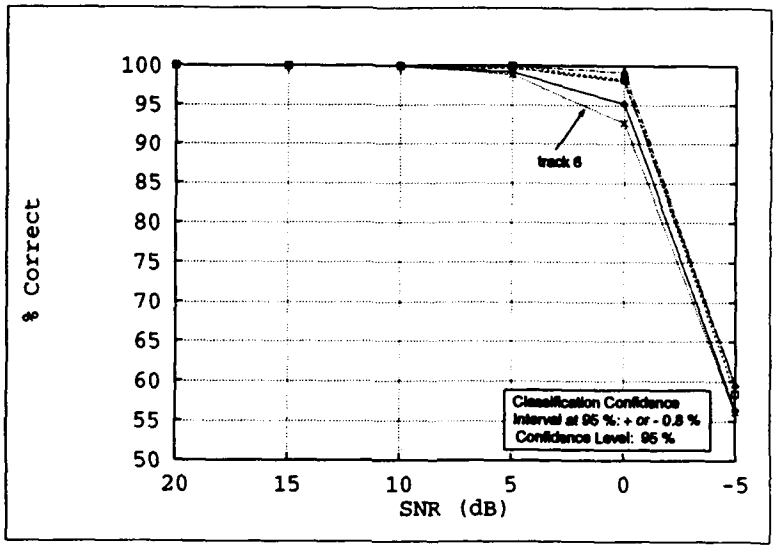


Figure 4.22. Single Look Multiple Range Profile Classification Rate (Tracks 2, 3, 5, 6, and 7)

4.6.2 Performance of the Sequential Vector Quantized Range Profile HMM Classifier As with the Single Look Classifier, M is chosen based on the average vector quantization error and the number of sequences required for training. Referring to Figure 4.9, the average quantization error decreases linearly as M increases. Although it is apparent that M should be larger than 1024, the number of training vectors required to train such a model would be prohibitive. For this classifier, M is chosen to be 128, based solely on the number of training vectors required. The HMM's are trained using the procedure outlined in Section 4.4.2.

Figure 4.23 shows the classification performance for tracks 1 and 4 at SNR's of -5 – 20 dB. The classification performance for the tracks not specifically trained on is shown in Figure 4.24. The performance is good for tracks 1 and 4 at SNR's of 20 dB and 15 dB, but for SNR's less than 15 dB the performance drops rapidly. With the exception of track 6, the classification for the tracks not in the training set is almost good as the performance for the tracks in the training set. Increasing the code book size (M) will probably improve the performance of this classifier, but the amount of training data required to train the HMM's may make the implementation of this type of classifier impractical.

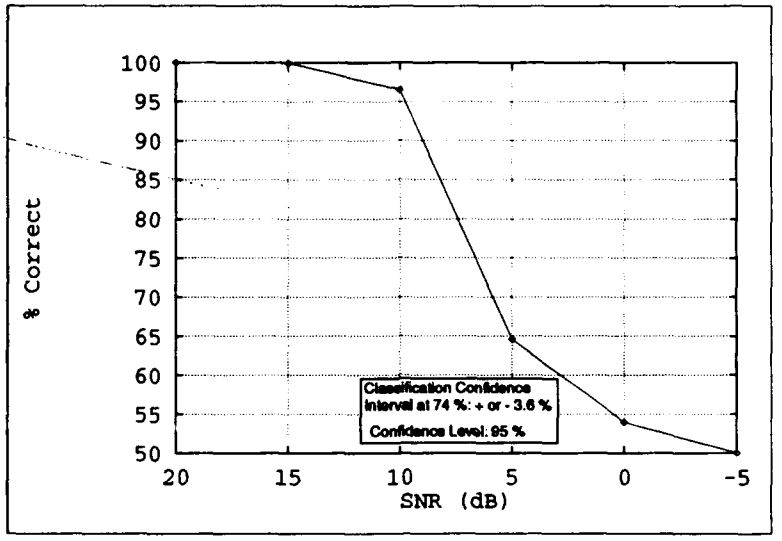


Figure 4.23. Classification Based on Sequences Vector Quantized Range Profiles (Tracks 1 and 4)

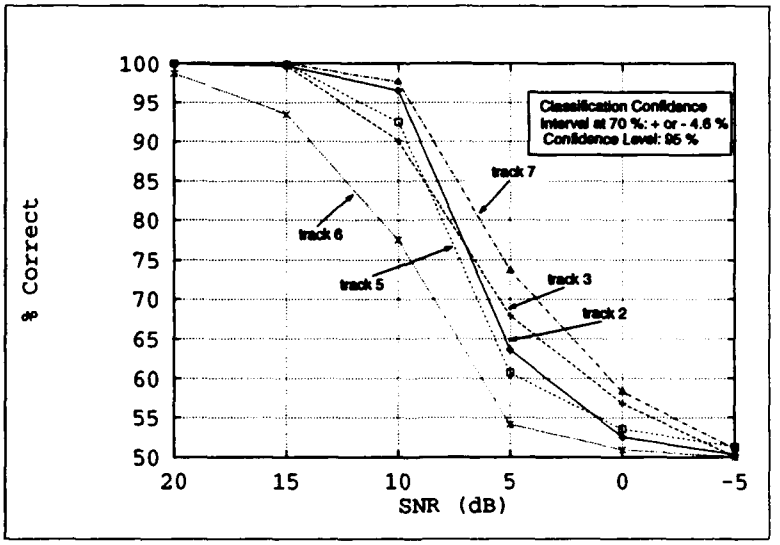


Figure 4.24. Classification Based on Sequences Vector Quantized Range Profiles (Tracks 2 , 3, 5, 6, and 7)

4.6.3 Performance of the Multiple State Sequential Range Profile Classifier Each of the 5 HMM's which correspond to the states of the Single Look HMM is itself a 5 state left-right HMM, but M is equal to 129 instead of 128. The extra symbol accounts for the null symbol (required to keep all state sequences the same length) which is discussed in Section 3.5.5. For this experiment the observation sequence length for states 1, 2 and 3 is limited (or extended if necessary by the null symbol) to 10 observations. The observation length for states 4 and 5 is set at 20. These observation lengths are determined empirically. That is, state 1 through 3 normally have observation sequence lengths of about 10 while the observation lengths for states 4 and 5 are normally about 20.

Each of the range profiles within these sequences is segmented into states using the Viterbi algorithm which is implemented using logarithms to avoid underflow. Again, the HMM's are trained using the procedure outlined in Section 4.4.2. Although this classifier is designed to use all 5 states, the best performance is realized when only states 2, 3, and 4 are used. Thus, the classification results reported here are based on the comparison of the log probabilities summed over states 2, 3, 4 instead of over all states.

Figure 4.25 shows the classification performance for track 1 and 4 at SNR's of -5 – 20 dB. The classification performance for the tracks not specifically trained on is shown in Figure 4.26. Consistent with previous results, the classification rate for track 6 is significantly worse than the other tracks. The performance of this classifier will probably improve if the training set is increased. However, the training set is limited by computing time. For 3,000 range profile sequences of 10 range profiles each, 30,000 range profiles must be processed. Essentially, this classifier, as well as the other sequential range profile classifiers, requires 10 times the number of training range profiles as the single look classifier.

4.6.4 Performance of the Uniform Multiple State Sequential Range Profile Classifier The HMM's for this classifier are all left-right models with $N = 5$ and $M = 128$. As described in Section 3.5.6, the range profiles are evenly segmented. Because each range profile sequence has 10 range profiles, each state contains 20 observations per range profile sequence. As shown in Figures 4.27 and 4.28, the performance of this classifier is almost identical to the performance of the Multiple State classifier. Again, track 6 has a classification rate that is significantly worse than the other tracks.

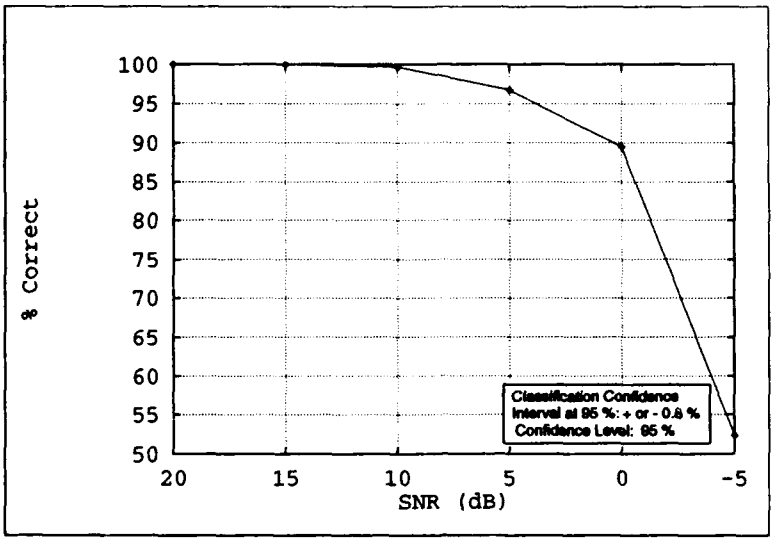


Figure 4.25. Multiple State Sequential Range Profile Classification Rate (Tracks 1 and 4)

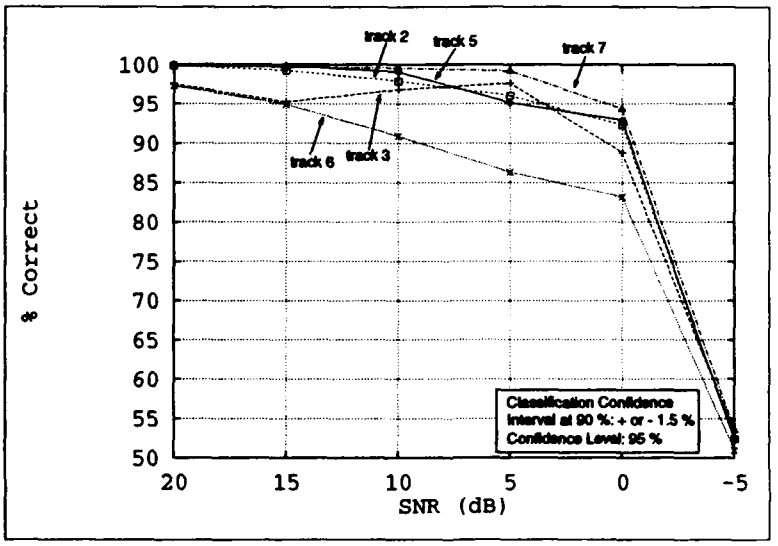


Figure 4.26. Multiple State Sequential Range Profile Classification Rate (Tracks 2, 3, 5, 6, and 7)

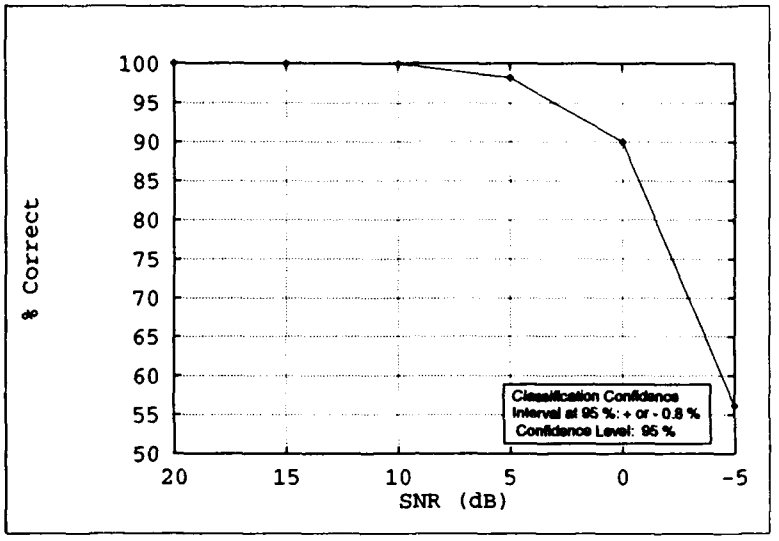


Figure 4.27. Uniform Multiple State Sequential Range Profile Classification Rate (Tracks 1 and 4)

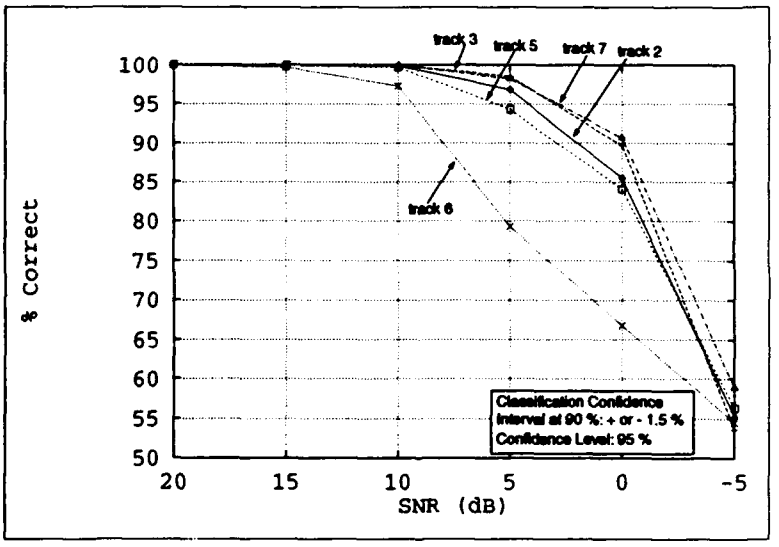


Figure 4.28. Uniform Multiple State Sequential Range Profile Classification Rate (Tracks 2, 3, 5, 6, and 7)

4.6.5 Effect of Range Profile Alignment on Overall Performance Figure 4.6.5 shows how each of the classifiers compare for tracks 1 and 4 when the range profiles are centered within the window. Figure 4.6.5 shows the performance of each of the classifiers when the range profile is allowed to vary in range within the range window with of Gaussian distribution (mean = 0 and variance = 2 m). The performance of all of the classifiers is reduced, but the overall performance is still very good. The classification performance for tracks 2, 3, 5, 6, and 7 shows similar sensitivity to variations in range.

4.7 Summary

In this chapter the experimental results which characterize the entire RTI process are presented. The front-end signal processing and feature extraction processing is verified using a known idealized range profile and typical Xpatch range profiles. The HMM training procedure is critical to the overall classification process. This procedure is verified by synthesizing HMM's using the training procedure from sequences that are generated with known models. The synthesized HMM's are almost identical the original HMM's used to generate the training sequences. The overall classification performance is excellent with the best results coming from the single look multiple range profile classifier when the range profile is centered in the range window. When the location of the range profile is allowed to vary within the range window, the performance of the Uniform Multiple State Sequential Range Profile classifier is about the same as the Single Look classifier.

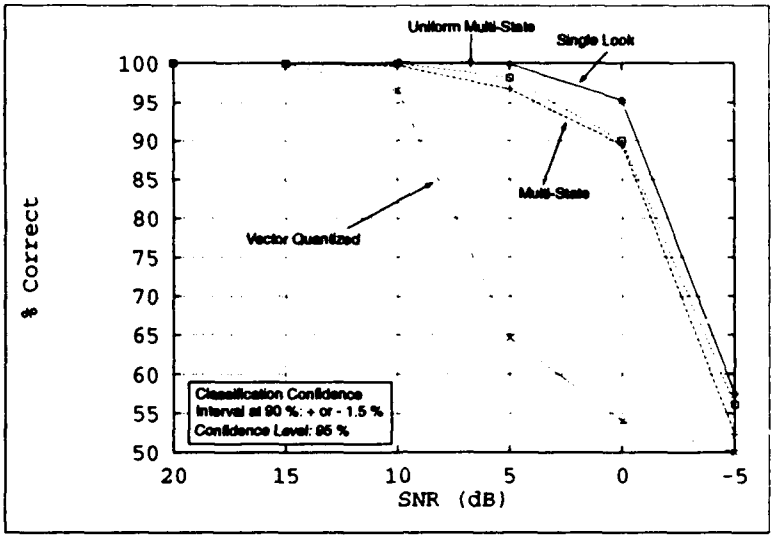


Figure 4.29. Performance of the Four Classifiers (Tracks 1 and 4) when the Range Profiles are Centered within the Range Window

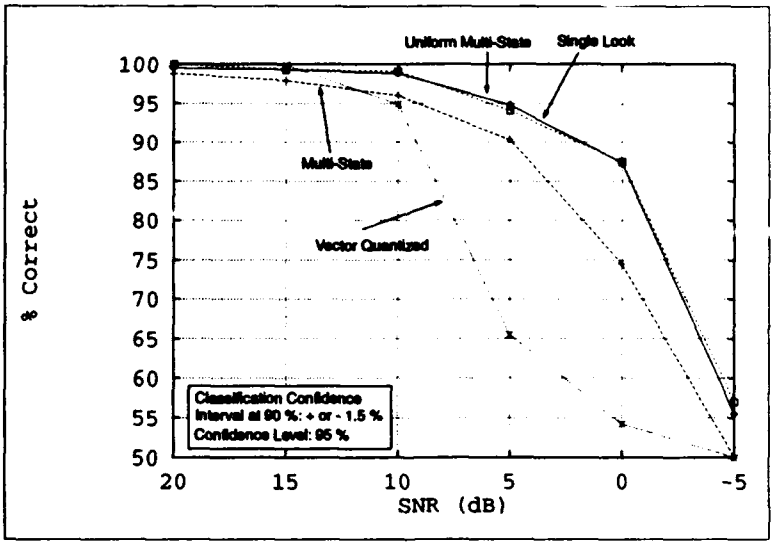


Figure 4.30. Performance of the Four Classifiers (Tracks 1 and 4) when the Range Profiles are not Centered within the Range Window

V. CONCLUSIONS AND RECOMMENDATIONS

5.1 Introduction

The goal of this research effort is to develop an RTI algorithm designed to identify aircraft using a Linear FM pulse compression HRR radar as a sensor. As a means of accomplishing this goal, the RTI process is divided into three sub-processes: front-end signal processing, feature extraction, and classification. The theory relating to these three sub-processes is presented in Chapter II and the Chapter III describes the RTI algorithms developed for this thesis by drawing upon the theory presented in Chapter II. The experimental procedures and results are given in Chapter IV. The purpose of this chapter is to provide the conclusion and recommendations based, in part, on these experimental results.

5.2 Conclusions

5.2.1 Front-end Signal Processing and Feature Extraction The purpose of the front-end signal processing is to emulate noise corrupted, circularly polarized HRR 'chirped' radar signatures. To validate this procedure, an idealized, known signature as well as typical Xpatch signatures are processed. These results, which are given in Section 4.2, compare well with theory. In short, the front-end signal processing is able emulate a physically realizable HRR 'chirp' radar. It should be noted, however, that this radar, which is simulated in software, is itself an ideal radar; free of all of the problems associated with implementing systems in hardware.

Based on the classification performance, which is discussed below, the feature set provided by the Prony Model describes the target well. However, at this point a near-real-time system could not be implemented using this feature extraction process because it simply takes too long to process the range profiles. The SVD in the parameter estimation procedure is the primary source of delay. The processing delay will be decreased if the bandwidth is decreased which in turn will also decrease the size of the matrix on which the SVD must be performed.

5.2.2 Classification Performance The classification algorithms developed for this thesis are based on the temporal relationships between: the individual scattering centers of HRR radar range profiles, the temporal relationships between range profiles, or both. One of the underlaying assumptions, behind all of these algorithms is that better classification performance will be realized if sequences of

observations are tied together in some logical way and not treated as individual events. The left-right HMM is proposed as a possible way to represent the temporal relationships between the observations in these sequences. The classification of any given HMM classifier is directly related to how well the parameters of the HMM's within the classifier are estimated by the training procedure. Therefore, the training procedure is a critical part of the RTI algorithms. The HMM training procedure and the performance of the classifiers developed for this thesis are discussed below.

5.2.2.1 HMM Training Clearly, the results presented in Section 4.4 show that the HMM training procedure is capable of estimating the model parameters to a high degree of accuracy. The significant finding here is that the model parameters are estimated with no *apriori* knowledge. The classification performance of all of the classification algorithms suggest that the training procedure is also able to accurately estimate the parameters of the HMM's used for classification.

5.2.2.2 Performance of the Single Look Classifier The Single Look classifier performs well, but at 20 dB SNR the KNN classifier is better. However, the Single Look classifier is more tolerant to noise. It should be noted that both the Single Look and KNN classification rate will approach 100 % if the size of the aspect angle window is decreased. Because the classification rate is well below 100 % for the 10° by 10° aspect angle window, the size of the aspect angle window represented by one HMM is essentially limited to this size. If more targets are added, the size of the window may have to be decreased further.

However, if the Single Look classifier waits to make the classification decision after a number of range profiles are processed, the classification performance is greatly improved. For sequences of 10 range profiles, the classification rate is almost 100 % for SNR down to 5 dB and about 95 % for a SNR of 0 dB. At a SNR of -5 dB, the classification performance drops dramatically. Thus, this classifier is good to about 0 dB SNR. In addition to this classifiers high tolerance for noise, it is also aspect angle track invariant. That is, it is not sensitive to how the target travels through the aspect angle window.

A major consideration in any HRR RTI system is range profile alignment and most of the classifiers in the literature are extremely sensitive to this alignment (17). However, the performance of this classifier does not decrease dramatically when the target is allowed to vary within the range

window. Therefore, if the tracking system is able to keep the target centered within the range window within a normal distribution of ± 2 m, this classifier will work without a centering process.

5.2.2.3 Performance of the Vector Quantized Sequential Range Profile Classifier The performance of this classifier is good for SNR of 10 dB and above, but drops rapidly below 10 dB. However, even though this classifier is trained on tracks 1 and 4, it is not overly sensitive to the aspect angle track. The degraded performance below 10 dB SNR is caused by a combination of less than adequate training and too much error induced by vector quantizing. The solution to these two problems is to increase the code book size and the training data set size. However, as discussed in Section 4.6.2, if enough training data is not available for a code book size of 128, the benefits of increasing the code book size to reduce the quantization error will be degraded by reducing the training set further. This classifier is also not overly sensitive to range alignment, but the degraded performance below 10 dB SNR limits its usefulness.

5.2.2.4 Performance of the Multiple State Sequential Range Profile Classifier Considering that this classifier accounts for both the temporal relationships between scattering centers of the individual range profiles as well as the temporal relationships between the range profile, this should have performed better the Single Look classifier which sees each range profile as an independent event. One possible explanation is that by normalizing the range profiles some of the information about the absolute RCS and how it changes with aspect angle is lost (4). In effect, the classification process is optimized for the Single Look classifier because the range profiles are normalized and therefore more correlated between aspect angles. It should be added that the classification rate for this classifier may improve if the size of the training set is increased. As with the Vector Quantized Sequential Look classifier, this classifier is not overly sensitive to the aspect angle track or range shifts.

5.2.2.5 Performance of the Uniform Multiple State Sequential Range Profile Classifier Although the performance of this classifier is slightly better when the range profiles are centered than the classifier discussed above, it is not statistically significantly better. However, this classifier is not as complex as the Multiple State Sequential Range Profile classifier because the dependence on the Single Look HMM's is eliminated. Moreover, when the locations of the range profiles are allowed to vary the performance of this classifier is better than the Multiple State Sequential Range Profile classifier and is

nearly the same as the Single Look classifier. Thus, the performance of this classifier is comparable to the Single Look classifier even though it is trained with a less than adequate training set.

5.3 Recommendations

In this thesis the problem is limited to two classes, but the performance of the Uniform Multiple State Sequential Range Profile classifier may be better than the Single Look classifier if the number of classes is increased. Thus, first and foremost the number of classes should be increased so that the advantages and limitations of the different classifiers can be better understood.

Secondly, the feature extraction processing time must be addressed by either improving the Prony Model parameter estimation procedure or reducing the amount of data which must be processed. Short of changing the Prony Model parameter estimation procedure, processing time can be reduced by decreasing the bandwidth. However, reducing the bandwidth also decreases the range resolution. So, there is a clear trade-off between processing time and bandwidth. Therefore, the relationship between bandwidth and classification performance should be investigated.

In this thesis, three 5 state left-right HMM classifiers are implemented, but there are hundreds of different HMM configurations which might be tried. Consequently, continued research should also investigate innovative ways of configuring the HMM's with the goal of finding a sequential range profile classifier with improved performance.

Finally, the human visual system recognizes objects based on motion, color, and form. These three feature sets are processed in separate areas of the brain before being merged to form a solution (32). Of course this explanation overly simplifies the way the brain does pattern recognition, but the point is that within the brain, identification is not based on one feature set alone. In this thesis, an attempt is made to use features derived from independent range profiles (form) and features derived from sequences of range profiles (motion) with good results. Perhaps the real solution to the HRR RTI problem as well as other target recognition problems will be resolved when more is understood about the brain and how it processes information.

Appendix A. DERIVATIONS OF IMPORTANT HMM VARIABLES

A.1 Introduction

In this appendix the derivations that apply to HMM will be presented. Specifically, the forward and backward algorithms, $\gamma_{t_n}(i)$, and $\xi_{t_n}(i,j)$ will be derived.

A.2 Derivation of the Forward Algorithm

The forward variable is defined as

$$\alpha_{t_n}(i) = P\left(O_{t_1}O_{t_2}\cdots O_{t_n}, q_{t_n} = S_i|\lambda\right) \tag{A.1}$$

or the joint probability of the partial observation sequence $O_{t_1}O_{t_2}\cdots O_{t_n}$ and being in state $S_i = q_{t_n}$ at time t_n, given the model λ (22) (21). $\alpha_{t_n}(i)$ is evaluated inductively as follows:

1. Initialization:

$$\begin{aligned}
\alpha_{t_1}(i) &= P\left(O_{t_1}, q_{t_1} = S_i|\lambda\right), && 1 \le i \le N \\
&= P\left(O_{t_1}|q_{t_1} = S_i, \lambda\right) P\left(q_{t_1} = S_i|\lambda\right), && 1 \le i \le N \\
&= b_i(O_{t_n})\pi_i, && 1 \le i \le N
\end{aligned} \tag{A.2}$$

2. Given $\alpha_{t_n}(i)$ for $1 \le i \le N$ calculate $\alpha_{t_{n+1}}(j)$ by induction:

$$\begin{aligned}
\alpha_{t_{n+1}}(j) &= P\left(O_{t_1}O_{t_2}\cdots O_{t_{n+1}}, q_{t_{n+1}} = S_j|\lambda\right), && 1 \le j \le N \quad t_1 \le t_n \le t_{t_{T-1}} \\
&= P\left(O_{t_{n+1}}|O_{t_1}O_{t_2}\cdots O_{t_n}, q_{t_{n+1}} = S_j, \lambda\right) \\
&\quad P\left(O_{t_1}O_{t_2}\cdots O_{t_n}, q_{t_{n+1}} = S_j|\lambda\right), && 1 \le j \le N \quad t_1 \le t_n \le t_{T-1} \\
&= b_j\left(O_{t_{n+1}}\right) P\left(O_{t_1}O_{t_2}\cdots O_{t_n}, q_{t_{n+1}} = S_j|\lambda\right), && 1 \le j \le N \quad t_1 \le t_n \le t_{T-1}
\end{aligned}$$

The last term on the right side can be expressed in terms of the known quantities a_{ij} and $\alpha_{t_n}(i)$.

$$\begin{aligned}
P\left(O_{t_1}O_{t_2}\cdots O_{t_n}, q_{t_{n+1}} = S_j|\lambda\right) &= \sum_{i=1}^{N} P\left(O_{t_1}O_{t_2}\cdots O_{t_n}, q_{t_{n+1}} = S_j, q_{t_n} = S_i|\lambda\right) \\
&= \sum_{i=1}^{N} P\left(q_{t_{n+1}} = S_j|O_{t_1}O_{t_2}\cdots O_{t_n}, q_{t_n} = S_i, \lambda\right)
\end{aligned}$$

$$P(O_{t_1}O_{t_2}\cdots O_{t_n}, q_{t_n} = S_i|\lambda)$$
$$= \sum_{i=1}^{N} a_{ij}\alpha_{t_n}(i)$$

It follows from direct substitution that

$$\alpha_{t_{n+1}}(j) = b_j(O_{t_{n+1}}) \sum_{i=1}^{N} a_{ij}\alpha_{t_n}(i), \quad 1 \le i \le N \qquad \text{(A.3)}$$
$$t_1 \le t_n \le t_{T-1}$$

3. The final step is the termination step in which the desired probability, $P(O|\lambda)$, is computed.

$$P(O_{t_1}O_{t_2}\cdots O_{t_T}|\lambda) = \sum_{i=1}^{N} P(O_{t_1}O_{t_2}\cdots O_{t_T}, q_{t_T} = S_i|\lambda)$$
$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_{t_T}(i) \qquad \text{(A.4)}$$

A.3 Derivation of the Backward Algorithm

The derivation of the backward algorithm is similar to the forward algorithm. The backward variable $\beta_{t_n}(i)$ is defined as

$$\beta_{t_n}(i) = P(O_{t_{n+1}}O_{t_{n+2}}\cdots O_{t_T}|q_{t_n} = S_i, \lambda). \qquad \text{(A.5)}$$

Hence, $\beta_{t_n}(i)$ is the probability of observing the partial observation sequence $O_{t_{n+1}}O_{t_{n+2}}\cdots O_{t_T}$, given state S_i and time t_n and the model λ (21) (22). Again the derivation proceeds inductively as follows:

1. Initialization: $\beta_{t_T}(i)$ is initialized to 1 for all i to maintain the desired probability. O is not explicitly defined for $t_n > t_T$, so the following probability is arbitrarily set to 1 (22).

$$\beta_{t_T}(i) = P(O_{t_T}\cdots O_{t_{T+1}}|q_{t_T} = S_i, \lambda) = 1, \quad 1 \le i \le N \qquad \text{(A.6)}$$

2. Induction: Compute β_{t_n} inductively using $\beta_{t_{n+1}}(i)$.

$$\beta_{t_n}(i) = P(O_{t_{n+1}}\cdots O_{t_T}|q_{t_n} = S_i, \lambda)$$

$$
\begin{aligned}
&= \sum_{j=1}^{N} P\left(O_{t_{n+1}} \cdots O_{t_T}, q_{t_{n+1}} = S_j | q_{t_n} = S_i, \lambda\right) \\
&+ \sum_{j=1}^{N} P\left(q_{t_{n+1}} = S_j | q_{t_n} = S_i, \lambda\right) P\left(O_{t_{n+1}} \cdots O_{t_T} | q_{t_{n+1}} = S_j, q_{t_n} = S_i, \lambda\right) \\
&= \sum_{j=1}^{N} a_{ij} P\left(O_{t_{n+1}} | O_{t_{n+2}} \cdots O_{t_T}, q_{t_{n+1}} = S_j, q_{t_n} = S_i, \lambda\right) \\
&\quad P\left(O_{t_{n+2}} \cdots O_{t_T} | q_{t_{n+1}} = S_j, q_{t_n} = S_i, \lambda\right) \\
&= a_{ij} b_j\left(O_{t_{n+1}}\right) P\left(O_{t_{n+2}} \cdots O_{t_T} | q_{t_{n+1}} = S_j, q_{t_n} = S_i, \lambda\right).
\end{aligned}
$$

By the Markov property, $O_{t_{n+2}} \cdots O_{t_T}$ is independent of $q_{t_n} = S_i$. Therefore,

$$
\beta_{t_n}(i) = a_{ij} b_j\left(O_{t_{n+1}}\right) P\left(O_{t_{n+2}} \cdots O_{t_T} | q_{t_{n+1}} = S_j, \lambda\right)
$$

Which is the desired relationship

$$
\beta_{t_n}(i) = a_{ij} b_j\left(O_{t_{n+1}}\right) \beta_{t_{n+1}}(j), \quad \begin{array}{l} t_{T-1} \geq t_n \geq t_1, \\ 1 \leq i \leq N. \end{array} \tag{A.7}
$$

3. Termination: The desired probability is computed as

$$
\begin{aligned}
P\left(O_{t_1} \cdots O_{t_T} | \lambda\right) &= \sum_{i=1}^{N} P\left(O_{t_1} \cdots O_{t_T}, q_{t_1} = S_i | \lambda\right) \\
&= \sum_{i=1}^{N} P\left(q_{t_1} = S_i | \lambda\right) P\left(O_{t_1} \cdots O_{t_T} | q_{t_1} = S_i, \lambda\right) \\
&= \sum_{i=1}^{N} \pi_i P\left(O_{t_1} | O_{t_2} \cdots O_{t_T}, q_{t_1} = S_i, \lambda\right) \\
&\quad P\left(O_{t_2} \cdots O_{t_T} | q_{t_1} = S_i, \lambda\right) \\
&= \sum_{i=1}^{N} \pi_i b_i\left(O_{t_1}\right) \beta_{t_1}(i)
\end{aligned} \tag{A.8}
$$

A.4 Derivation of $\gamma_{t_n}(i)$

$$
\gamma_{t_n}(i) = P\left(q_{t_n} = S_i | O_{t_1} \cdots O_{t_n} O_{t_{n+1}} \cdots O_{t_T}, \lambda\right)
$$

$$= \frac{P(O_{t_1} \cdots O_{t_n} O_{t_{n+1}} \cdots O_{t_T} | q_{t_n} = S_i, \lambda)\, P(q_{t_n} = S_i | \lambda)}{\sum_{j=1}^{N} P(O | q_{t_n} = S_j, \lambda)\, P(q_{t_n} = S_j | \lambda)}$$

$$= \frac{P(O_{t_1} \cdots O_{t_n} | q_{t_n} = S_i, \lambda)\, P(q_{t_n} = S_i | \lambda)\, P(O_{t_{n+1}} \cdots O_{t_T} | q_{t_n} = S_i, \lambda)}{\sum_{j=1}^{N} P(O | q_{t_n} = S_j, \lambda)\, P(q_{t_n} = S_j | \lambda)}$$

The last factor in the numerator on the right side is, by definition, $\beta_{t_n}(i)$. The remaining factors in the numerator are equal to $\alpha_{t_n}(i)$ as is shown below.

$$P(O_{t_1} \cdots O_{t_n} | q_{t_n} = S_i, \lambda)\, P(q_{t_n} = S_i | \lambda) = \left(\frac{P(O_{t_1} O_{t_2} \cdots O_{t_n}, q_{t_n} = S_i | \lambda)}{P(q_{t_n} = S_i | \lambda)} \right) P(q_{t_n} = S_i | \lambda)$$

$$= \alpha_{t_n}(i)$$

And similarly, the denominator is equal to $P(O|\lambda)$, such that

$$\sum_{j=1}^{N} P(O | q_{t_n} = S_j, \lambda)\, P(q_{t_n} = S_j | \lambda) = \left(\frac{P(O, q_{t_n} = S_j | \lambda)}{P(q_{t_n} = S_j | \lambda)} \right) P(q_{t_n} = S_j | \lambda)$$

$$= \sum_{j=1}^{N} \alpha_{t_T}(j)$$

$$= \sum_{j=1}^{N} \alpha_{t_n}(j) \beta_{t_n}(j)$$

$$= P(O|\lambda)$$

Now $\gamma_{t_n}(i)$ is expressed in terms of $\alpha_{t_n}(i)$ and $\beta_{t_n}(i)$ so that

$$\gamma_{t_n}(i) = \frac{\alpha_{t_n}(i) \beta_{t_n}(i)}{P(O|\lambda)} \tag{A.9}$$

A.5 Derivation of $\xi_{t_n}(i,j)$

$$\xi_{t_n}(i,j) = \frac{P(O, q_{t_n} = S_i, q_{t_{n+1}} = S_j | \lambda)}{P(O | \lambda)}$$

$$= \frac{P(O_{t_1} \cdots O_{t_n}, q_{t_n} = S_i, q_{t_{n+1}} = S_j|\lambda)\, P(O_{t_{n+1}} \cdots O_{t_T}, q_{t_n} = S_i, q_{t_{n+1}} = S_j|\lambda)}{P(O|\lambda)}$$

In first term the numerator the joint probability of $O_{t_1} \cdots O_{t_n}$ and $q_{t_n} = S_i$ is independent of $q_{t_{n+1}} = S_j$, so

$$\begin{aligned} P(O_{t_1} \cdots O_{t_n}, q_{t_n} = S_i, q_{t_{n+1}} = S_j|\lambda) &= P(O_{t_1} \cdots O_{t_n}, q_{t_n} = S_i|\lambda)\, P(q_{t_{n+1}} = S_j|\lambda) \\ &= \alpha_{t_n}(i) P(q_{t_{n+1}} = S_j|\lambda) \end{aligned}$$

The second term in the numerator can be rearranged so that

$$\begin{aligned} P(O_{t_{n+1}} \cdots O_{t_T}, q_{t_n} = S_i, q_{t_{n+1}} = S_j|\lambda) &= P(O_{t_{n+2}} \cdots O_{t_T}|O_{t_{n+1}}, q_{t_n} = S_i, q_{t_{n+1}} = S_j, \lambda) \\ &\quad P(O_{t_{n+1}}, q_{t_n} = S_i, q_{t_{n+1}} = S_j|\lambda) \\ &= P(O_{t_{n+2}} \cdots O_{t_T}|q_{t_{n+1}} = S_j, \lambda)\, P(O_{t_{n+1}}, q_{t_n} = S_i, q_{t_{n+1}} = S_j|\lambda) \\ &= \beta_{t_{n+1}}(j) P(O_{t_{n+1}}, q_{t_n} = S_i, q_{t_{n+1}} = S_j|\lambda) \end{aligned}$$

The second equality holds because $O_{t_{n+2}} \cdots O_{t_T}$ is independent of $O_{t_{n+1}}$ and $q_{t_n} = S_i$. It follows that

$$\begin{aligned} \xi_{t_n}(i,j) &= \frac{\alpha_{t_n}(i)\beta_{t_{n+1}}(j) P(O_{t_{n+1}}, q_{t_n} = S_i, q_{t_{n+1}} = S_j|\lambda)\, P(q_{t_{n+1}} = S_j|\lambda)}{P(O|\lambda)} \\ &= \frac{\alpha_{t_n}(i)\beta_{t_{n+1}}(j) P(O_{t_{n+1}}, q_{t_n} = S_i, q_{t_{n+1}} = S_j|\lambda)}{P(O|\lambda)} \\ &= \frac{\alpha_{t_n}(i)\beta_{t_{n+1}}(j) a_{ij} b_j(O_{t_{n+1}})}{P(O|\lambda)} \end{aligned}$$

Appendix B. XPATCH DATA FORMAT

B.1 Introduction

In this appendix Xpatch data format is outlined. Xpatch is written in FORTAN, therefore the data format is described in those terms. The information contained in this appendix was supplied by Wright-Labs/AARA.

B.2 Output File Format

The output file is an unformatted direct access file with a record length of 32 bytes. The first several records contain the simulation configuration parameters and the subsequent records contain the complex polarimetric frequency samples. The number of the header records depend on the number of nonuniform frequency samples as described in the following paragraph. The sample FORTRAN open statement shown for a SUN workstation has a record length given by the number of bytes. For other platforms, such as SGI and VAX, the record length is specified in the number of 4 byte words and has a value of 8.

```
OPEN( UNIT   = lundb,
:     FILE   = filnam,
:     ACCESS = 'direct',
:     FORM   = 'unformatted',
:     STATUS = 'old',
:     RECL   = 32)
```

The variables contained in the header records can be read as shown with the variable declarations and descriptions given by Table B1. The bottom of the code shows how to read the nonuniform frequency samples and assumes that an array freqSample has been declared with maxFreqSamples elements.

```
READ(lundb, REC = 1) simTitle(1:32)
READ(lundb, REC = 2) simTitle(33:64)
```

```
      READ(lundb, REC = 3) modelTitle(1:32)
      READ(lundb, REC = 4) modelTitle(33:64)
      READ(lundb, REC = 5) (simDate(i), i=1,3), (modelData(i), i=1,3),
     :                     snglOpt, edgeOpt
      READ(lundb, REC = 6) multOpt,  rayDensity,
     :                     zoneCellSize, subdivFac,
     :                     numMaxBounce, angStartTgtAz,
     :                     angStartTgtEl, numSigProfileAz
      READ(lundb, REC = 7) numSigProfileEl, angSpacingAz,
     :                     angSpacingEl, freqSpacingType,
     :                     freqStart, numFreqSamples,
     :                     freqSpacingInc, numHeadersLeft

      IF (freqSpacingType .EQ. 2) THEN
         IF (numFreqSamples .GT. maxFreqSamples) THEN
            WRITE(0,*)'Number of frequency samples greater than ',
     :              'array dimension'
            WRITE(0,*)'Number of frequency samples = ',numFreqSamples
            WRITE(0,*)'Array freqSample dimension  = ',maxFreqSamples
            STOP
         ENDIF

         freqSample(1) = freqStart
         IF (numFreqSamples .GT. 1) THEN
            numrec = (numFreqSamples - 2) / samplesPerRecord + 1

            DO 10 irec=1,numrec
               istart = (irec - 1) * samplesPerRecord + 2
               iend   = min(istart+samplesPerRecord-1, numFreqSamples)
               READ(lundb,REC=nxtrec) (freqSample(i), i=istart,iend)
```

```
            nxtrec = nxtrec + 1
10        CONTINUE

      ENDIF

   ENDIF
```

Table B.1. Xpatch Output File Header Variables

Variable Name	Variable Type	Variable Description
simTitle	character*64	title of simulation run
modelTitle	character*32	title of model file
simDate	5 * integer*4	date of simulation run
modelDate	5 * integer*4	creation date of model file
snglOpt	integer*4	single bounce option
edgeOpt	integer*4	edge diffraction option
multOpt	integer*4	multiple bounce option
rayDensity	real*4	incident ray density
zoneCellSize	real*4	zone cell size for ray processing
subdivFac	real*4	ray tracer subdivision factor
numMaxBounce	integer*4	maximum number of bounces allowed
angStartTgtAz	real*4	starting target azimuth angle
angStartTgtEl	real*4	starting target elevation angle
numSigProfileAz	integer*4	number of azimuth signature profiles
numSigProfileEl	integer*4	number of elevation signature profiles
angSpacingAz	real*4	azimuth angular spacing increment
angSpacingEl	real*4	elevation angular spacing increment
freqSpacingType	integer*4	sampling type (0=uniform,1=nonuniform)
freqStart	real*4	illumination starting frequency
freqSpacingInc	real*4	uniform frequency spacing
numHeadersLeft	integer*4	number of headers remaining in file

The frequency data is written to the output file with the contents of each record corresponding to four complex double precision numbers, using complex*8 representation. The first complex number in each record represents the VV polarization, followed by VH, HV, and HH polarizations. The data records increment according to increasing frequency. A set of numFreqSamples records corresponds to a single signature profile. The first signature profile corresponds to an orientation described by the

target azimuth and elevation start angles, angStartTgtAz and angStartTgtEl, respectively. An azimuth angle of zero represents illumination from nose on and increasing azimuth is clockwise when viewed from above the target. The target elevation is defined as zero in the plane of the target and a positive value indicates a view angle from above the target (ARTI convention).

Multiple signature profiles within an output file are controlled by the angular spacing in the azimuth and elevation directions, angSpacingAZ and angSpacingEl, and by the number of signature profiles in the azimuth and elevation directions, numSigProfileAz and numSigProfileEl. The order of signature profiles increment in the azimuth direction followed by the elevation direction. An example using FORTRAN to read the frequency data is shown.

```
COMPLEX*8 vv, vh, hv, hh
irec = 7
DO i = 1, numSigProfileEl
   DO j = 1, numSigProfileAz
      DO k = 1, numFreqSamples
         irec = irec + 1
         READ(lundb, REC = irec) vv, vh, hv, hh
      END DO
    END DO
END DO
```

B.3 Utilities

The Utilities directory under xpatch contains routines to examine the xpatch output file and perform an FFT of frequency samples in order to create a range profile. The routine dumpSP has the following syntax:

```
dumpSP infile [azAngle] [elAngle] [numSig] [outfile]
```

where

infile - name of xpatch output file

azAngle - azimuth angle of starting profile [default = 0.0]

elAngle - elevation angle of starting profile [default = 0.0]

numSig - number of consecutive signature profiles dumped [default = 1]

outfile - formatted output file [default = screen].

Appendix C. SOURCE CODE

C.1 Introduction

This appendix contains the source code for the subroutines that are used for the Forward-Backward Algorithm, the Viterbi Algorithm with logrithms, and the Baum-Welch reestimation algorithm.

C.2 Forward-Backward Algorithm

The source code for the Forward-Backward algorithm follows the notation as presented in Section 2.4.12.1.

```
/*********************************************************
* Subroutine:  Forward/Backward Algorithm
* Date: 15 July 1992
* Author: Capt Mark DeWitt
*********************************************************/

#include <stdio.h>
#include <math.h>
/*these functions can be found in Numerical Recipies for C*/
float ***tensor();
float **matrix();
float *vector();
int **imatrix();

void fwdbck(alpha, beta, C, P, a, b, pi,O, M, N, K, T)
float ***alpha, ***beta, *P, **a, **b, **C, *pi;
int **O, M, N, K, T;

/* input/ouput variable discription

inputs:
N: number of states
M: number of observation symbols
T: number of observations per observation sequence
K: number of observation sequences
O: observations (KxT int matrix with first index: 1)
pi: initial state probabilities (Nx1 float vector first index: 1)
a: state transition probabilities (NxN float matrix first index: 1)
b: observation probabilities (NxM float matrix first index: 1)

outputs:
alpha: forward variable coefficients (KxNxT float tensor first index: 1)
beta: backward variable coefficients (KxNxT float tensor first index: 1)
```

```
C: scaling cefficients (KxT float matrix first index: 1)
P: observation probabilities (Kx1 float vector first index: 1) */

{
  int i, j, k, m, t;
  float prob;

/*Calculate the Scaled Alpha's and Beta's for all K observations using the Foward-Backward
Algorithms*/

for(k=1; k≤K; k++){

/*Calculate the alpha's for the k^th observation sequence*/
  for(t=1; t≤T; t++){
    C[k][t] = 0.0;
  if (t==1){
    for(i=1; i≤N; i++){/*initialize*/
    C[k][t] += alpha[k][i][t] = pi[i]*b[i][O[k][t]];} /*end i*/
  } /*end if*/
   else{
    for(j=1; j≤N; j++){/*recursion*/
    for(i=1, prob = 0.0; i≤N; i++){
        prob += a[i][j]*alpha[k][i][t−1];} /*end i*/
    C[k][t] += alpha[k][j][t] = prob*b[j][O[k][t]];} /*end j*/
    } /*end else*/

/* Scaling Factor */
   C[k][t] = 1/C[k][t];

/*Scale alpha's*/
   for(i=1; i ≤ N; i++){
    alpha[k][i][t] = C[k][t]*alpha[k][i][t];} /* end i*/
  } /*end t*/

/*Calculate the probability of the k^th observation sequence*/
  for(t = 1, P[k] = 0.0; t ≤ T; t++){
  P[k] += − (float)log10((double)C[k][t]);}

/*Calculate the beta's for the k^th observation sequence*/
  for(t=T; t≥1; t−−){
    if(t == T){
   for(i = 1; i ≤ N; i++){ /*initialize */
    beta[k][i][t] = 1.0 * C[k][t];}
  } /*end if */
    else{
   for(i = 1; i ≤ N; i++){
      for(j = 1, prob = 0.0; j ≤ N; j++){ /*recursion*/
        prob += beta[k][j][t+1]*a[i][j]*b[j][O[k][t+1]];} /*end j*/
    beta[k][i][t] = prob *C[k][t] ;} /*end i*/
    } /* end else */
  } /* end t*/
```

```
} /* end k and end fwd-bck calculations */
} /*end fwdbck */
```

C.3 Viterbi Algorithm with Logrithms

The source code for the Viterbi algorithm follows the notation presented in Section 2.4.12.2.

```
/*********************************************************
* Subroutine:  Viterbi Algorithm
* Date: 15 July 1992
* Author: Capt Mark DeWitt
*********************************************************/

#include <stdio.h>
#include <math.h>
/*these functions can be found in Numerical Recipies for C*/
float ***tensor();
float **matrix();
float *vector();
int **imatrix();
int *ivector();
void free_vector();

void viterbi_log(Q, Pvit, a, b, pi, O, M, N, K, T)
float  **a, **b, *pi, *Pvit;
int  **O, M, N, K, T, **Q;

/* input/ouput variable discription

inputs:
N: number of states
M: number of observation symbols
T: number of observations per observation sequence
K: number of observation sequences
O: observations (KxT  int matrix with first index: 1)
pi:  initial state probabilities (Nx1 float vector first index: 1)
a:  state transition probabilities (NxN float matrix first index: 1)
b:  observation probabilities  (NxM float matrix first index: 1)

outputs:
Q: most probable state sequence (KxT  int matrix with first index: 1)
Pvit:  probability of the best state sequence  (Kx1 float vector first index: 1) */

\*NOTE:  indxx(x ,y, z) is a sorting routine (see Numerical Recipies for C), where x is the number of
elements in the vector y to be sorted.  z (ouput of indxx) is an int vector containing the indicies of the
elements in y in assending order.*/
{
int i, j, k, m, t;
float **delta, *temp;
int **psi, *indx;
```

```
/* allocate memory for the local variables */
delta=matrix(1,N,1,T);
psi=imatrix(1,N,1,T);
temp=vector(1,N);
indx=ivector(1,N);

/*  set all zero probabilities to 1e-20 */
for(i = 1; i ≤ N; i++){
   if(pi[i] == 0) pi[i] = 1e−20;
   for(j = 1; j ≤ N; j++) if(a[i][j] == 0.0) a[i][j] = 1e−20;
}



for(k = 1, Pvit[k] = 0.0; k ≤ K; k++){

   for(t = 1; t ≤ T; t++){
      if(t == 1){
        for(i = 1; i ≤ N; i++){
              delta[i][t] = (float)(log10((double)pi[i])  + log10((double)b[i][O[k][t]]));
          psi[i][t] = 0;} /* end t */
     } /*end if */
          else{

        for(j = 1; j ≤ N; j++){
        for(i = 1; i ≤ N; i++)
                  temp[i] = delta[i][t−1] +(float)log10((double)a[i][j]);
              indexx(N, temp, indx);
              psi[j][t] = indx[N];
              delta[j][t] = temp[indx[N]] + (float)log10((double)b[j][O[k][t]]);
             } /* end j */
          } /*end else */
   } /* end t */

   for(t = T; t ≥ 1; t−−){
      for(i = 1; i ≤ N; i++) temp[i] = delta[i][t];
           indexx(N, temp, indx);
           if(t == T) {Q[k][t] = indx[N];}
      else {Q[k][t] = psi[Q[k][t+1]][t+1];}
       } /*end t */

   for(i = 1; i ≤ N; i++) temp[i] = delta[i][T];
   indexx(N, temp, indx);
      Pvit[k] = delta[indx[N]][T];
} /* end k */

free_vector(temp, 1,N);
free_ivector(indx, 1,N);
free_matrix(delta, 1,N,1,T);
free_imatrix(psi,1,N,1,T);
}
```

C.4 Buam-Welch Reestimation Algorithm

The source code for the Baum-Welch reestimation algorithm follows the notation presented in Section 3.5.1.

```
/*********************************************************
* Subroutine:  Buam-Welch Reestimation Algorithm
* Date: 15 July 1992
* Author: Capt Mark DeWitt
*********************************************************/

#include <stdio.h>
#include <math.h>
/*these functions can be found in Numerical Recipies for C*/
float ***tensor();
float **matrix();
float *vector();
int **imatrix();
void free_matrix();

void baum(alpha, beta, C, a, b, abar, bbar, pi, pibar,O, M, N, K, T)

float ***alpha, ***beta,  **a, **b, **C, *pi,**abar, **bbar, *pibar;
int **O, M, N, K, T;
/* input/ouput variable discription

inputs:
N: number of states
M: number of observation symbols
T: number of observations per observation sequence
K: number of observation sequences
O: observations (KxT  int matrix with first index: 1)
pi:  initial state probabilities (Nx1 float vector first index: 1)
a:  state transition probabilities (NxN float matrix first index: 1)
b:  observation probabilities  (NxM float matrix first index: 1)
alpha:  forward variable coefficients (KxNxT float tensor first index: 1)
beta:  backward variable coefficients (KxNxT float tensor first index: 1)
C:  scaling cefficients (KxT float matrix first index: 1)

outputs:
pi:  updated initial state probabilities (Nx1 float vector first index: 1)
a:  ubdated state transition probabilities (NxN float matrix first index: 1)
b:  ubdated observation probabilities  (NxM float matrix first index: 1)*/

{
int i, j, k, m, t, tau;
double numk, num, denk, den;
float  prob;

/* update estimate of pi's */
```

```
for(i = 1; i ≤ N; i++){
   for(k = 1, numk = 0.0, denk = 1e−200; k ≤ K; k++){
      numk += (alpha[k][i][1]*beta[k][i][1]);
            for(j = 1, den = 0.0; j ≤ N; j++)
              den += (alpha[k][j][1]*beta[k][j][1]);
      denk += den;
   }
   pibar[i] = (float)(numk/denk);
}

/* update estimate of a's, Equation (3.6)*/

for(i = 1; i≤N; i++){
  for (j = 1; j ≤ N; j++){
       for(k = 1, numk = 0.0, denk = 1e−200; k ≤ K; k++){
      for(t = 1, num = 1e−200, den = 1e−200; t ≤ T−1; t++){
         num += alpha[k][i][t]*a[i][j]*b[j][O[k][t+1]]*beta[k][j][t+1]/C[k][t];
      den += alpha[k][i][t]*beta[k][i][t]/(C[k][t]*C[k][t]);
       } /*end t*/
          numk += num;
          denk += den;
        } /* end k */
   abar[i][j] = (float)(numk/denk);
   } /* end j */
} /* end i */

/* update estimate of b's, Equation (3.7) */

for(j = 1; j ≤ N; j++){
  for(m = 1; m ≤ M; m++){
   for(k = 1, numk = 0.0, denk = 1e−200; k ≤ K; k++){
        for(t = 1, num = 1e−200, den = 1e−200; t ≤ T; t++){
      if(m == O[k][t]) {num += alpha[k][j][t]*beta[k][j][t]/(C[k][t]);}
      den += alpha[k][j][t]*beta[k][j][t]/(C[k][t]);
      } /* end t */
        numk += num;
     denk += den;
       } /* end k */
   bbar[j][m] = (float)( numk/ denk);
   if(bbar[j][m] < 1e−10) bbar[j][m] = 1e−10;
   } /* end m */
} /* end j */
}
```

Bibliography

1. Beckner, F. *et. al. Automatic Radar Target Identification (ARTI) Phase II (U).* Technical Report WRDC-TR-90-1003, Wright-Patterson AFB, OH: General Dynamics (Pomona) and U.S. Air Force Wright Labs, General Dynamics, March 1992.
2. Chamberlain, Neil F., *et. al.* "Radar Target Identification of Aircraft Using Polarization-Diverse Features," *IEEE Transactions on Aerospace and Electronic Systems*, *27*(1):58–67 (January 1991).
3. Christensen, M. *er. al. Computing Range Profiles with Xpatch.* Technical Report ARTI-92-2, University of Illinois, January 1992.
4. Cohen, Marvin N. "Variability of Ultra-High Range Resolution Radar Profiles and Some Implications for Target Recognition," *SPIE*, *1699*:256–266 (1992).
5. Cook, Charles E. "Pulse Compression – Key to More Efficient Radar Transmission," *Proceedings of the IRE* (1960).
6. Davenport, Wilbur B, Jr. *Probability and Random Processes.* New York: McGraw-Hill Book-Company, 1987.
7. E., Baum Leonard. "An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes," *Inequalities*, *3*:1–8 (1972).
8. Farhat, Nabil H. "Microwave Diversity Imaging and Automatic Target Identification Based on Models of Neural Networks," *Proceedings of the IEEE*, *77*(5):670–680 (May 1989).
9. Gagliardi, Robert M. *Introduction to Communications Engineering* (Second Edition). New York: Jown Wiley & Sons, 1988.
10. Geurts, James F. *Target Recognition Using Surface Vibration Measurements.* MS thesis, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH 45433, December 1992.
11. H., Juang B. and Lawrence R. Rabiner. "A Probabilistic Distance Measure for Hidden Markov Models," *AT&T Technical Journal*, *64*(2):391–408 (February 1985).
12. Juang, B. H. "On the Hidden Markov Model and Dynamic Time Warping for Speech Recognition – A Unified View," *AT&T Bell Laboratories Technical Journal*, *63*(7):1213–1243 (September 1984).
13. Juang, Biing-Hwang and Lawrence R. Rabiner. "Mixture Autoregressive Hidden Markov Models for Speech Signals," *IEEE Transactions on Acoustic, Speech, and Signal Processing*, *33*(6):1404–1413 (December 1985).
14. Kruas, John D. *Electromagnetics* (third Edition). New York: McGraw-Hill Publishing Company, 1981.
15. Levinson, Stevin E., *et. al.* "An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition," *Bell Systems Technical Journal*, *62*(4):1035–1074 (April 1983).
16. Libby, Edmund W. *LTC, USA.* Personal Interview, AFIT/ENG, Wright-Patterson AFB OH 45433, 28 September 1992.

17. Libby, Edmund W. *Multisensor Data Fusion and Target Recognition.* Ph. D. Prospectus: Air Force Institute of Technology, Wright-Patterson AFB, OH 45424, 1992.

18. McGillem, Clare D. and George R. Cooper. *Continuous and Discrete Signal and System Analysis* (Second Edition). New York: Holt, Rinehart and Winston, 1984.

19. Parsons, Thomas W. *Voice and Speech Processing.* New York: McGraw-Hill Book Company, 1987.

20. Rabiner, Lawrence R., *et. al.* "On the Application of Vector Quantization and Hidden Markov Models to Speaker-Independent, Isolated Word Recognition," *The Bell Systems Technical Journal, 4*(4):1075–1105 (April 1983).

21. Rabiner, Lawrence R. "An Introduction to Hidden Markov Models," *IEEE ASSP Magazine*, 4–16 (January 1986).

22. Rabiner, Lawrence R. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE, 77*(2):257–285 (February 1989).

23. Rahman, M. A. and Yu K.-B. "Total least squares approach for frequency estimation using liera prediction," *IEEE Transactions on Acoustic, Speech, and Signal Processing, 35*:1440–1454 (October 1987).

24. Ruck, Dennis W. *Characterization of Multilayer Perceptrons and their Application to Multisensor Automatic Target Detection.* PhD dissertation, School of Engineering, AFIT WrightAir Force Institute of Technology, Wright-Patterson, 1990.

25. Sacchini, Joseph J. *Development of Two-Dimensional Parametric Radar Signal Modeling and Estimation Techniques with Application to Target Identification.* PhD dissertation, Ohio State University, Columbus Ohio, 1992.

26. Skolnik, Merrill I. *Introduction to Radar Systems* (Second Edition). New York: McGraw-Hill Publishing Company, 1980.

27. Steedly, William M. and Randolph L. Moses. "High Resolution Exponential Modeling of Fully Polarized Radar Returns," *IEEE Transactions on Aerospace and Electronic Systems, 27*(3):459–469 (May 1991).

28. Unkown, Author. *Automatic Radar Target Identification (ARTI) Phase IIIA: Algorithm Q User's Manual.* Technical Report contract F33615-89-C-1099, Item 0001, CDRL 018-A1D, General Dynamics, September 1991.

29. Van Syl, John J. *et. al.* "Imaging Radar Polarimetry from Wave Synthesis," *Proceedings of IGARSS*, 709–713 (August 1986).

30. von Schlachta, K. "A Contribution to Radar Target Classification," *Proceedings of the IEEE, 60*(2):135–139 (May 1980).

31. Xu Lei, *et. al.* "Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition," *IEEE Transactions on Systems, Man, and Cybernetics, 22*(3):418–435 (May/June 1992).

32. Zeki, Semir. "The Visual Image in Mind and Brain," *Scientific American*, 69 – 76 (September 1992).

Vita

Captain Mark R. DeWitt was born on October 24, 1958 in Madison, Wisconsin. Captain DeWitt graduated from Beatrice High School in Beatrice, Nebraska in May 1977. He entered the Air Force in August 1977 and served as an Avionics Navigation Systems Technician and Field Training Instructor at Dyess AFB, Texas; Incirlik AB, Turkey; and Bergstrom AFB, Texas. In September 1984, he attended the University of Texas at Austin under the Airmans Education and Commissioning Program. He graduated from the University of Texas with a Bachelors of Science Degree in Electrical Engineering in May 1987. Upon graduation, he attended Officers Training School and was commissioned into the USAF in September 1987. Captain DeWitt was then assigned to the Headquarters Electronic Security Command, Kelly AFB, Texas where he served as an Operational Test and Evaluation Program Manager, from September 1987 to May 1991. In June 1991, he entered the School of Engineering, Air Force Institute of Technology at Wright-Patterson AFB, Ohio, to pursue a Master of Science Degree in Electrical Engineering with an emphasis in communications and radar systems technology. He is married to Rita (Coyle) DeWitt of Dallas, Texas and they have three children; Robin, Jonathan and Cassie.

Permanent address: 4938 Timber Whip Dr.
San Antonio, Texas 28250

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1992	Master's Thesis

4. TITLE AND SUBTITLE
HIGH RANGE RESOLUTION RADAR TARGET IDENTIFICATION USING THE PRONY MODEL AND HIDDEN MARKOV MODELS

5. FUNDING NUMBERS

6. AUTHOR(S)
Mark R. DeWitt

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GE/92D-15

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Mr. Ed Zelinio
Wight Laboratories/AARA
WPAFB OH 45433-6583

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Fully polarized Xpatch signatures are transformed to two left circularly polarized signals. These two signals are then filtered by a linear FM pulse compression ('chirp') transfer function, corrupted by AWGN, and filtered by a filter matched to the 'chirp' transfer function. The bandwidth of the 'chirp' radar is about 750 MHz. Range profile feature extraction is performed using the TLS Prony Model parameter estimation technique developed at Ohio State University. Using the Prony Model, each scattering center is described by a polarization ellipse, relative energy, frequency response, and range. This representation of the target is vector quantized using a K-means clustering algorithm. Sequences of vector quantized scattering centers as well as sequences of vector quantized range profiles are used to synthesize target specific Hidden Markov Models (HMM's). The identification decision is made by determining which HMM has the highest probability of generating the unknown sequence. The data consist of synthesized Xpatch signatures of two targets which have been difficult to separate with other RTI algorithms. The RTI algorithm developed for this thesis is clearly able to separate these two targets over a 10 by 10 degree (1 degree granularity) aspect angle window off the nose for SNRs as low as 0 dB. The classification rate is 100 % for SNRs of 5 – 20 dB, 95 % for a SNR of 0 dB and it drops rapidly for SNRs lower than 0 dB.

14. SUBJECT TERMS	15. NUMBER OF PAGES
High Range Resolution RADAR, Prony Model, Hidden Markov Models, Linear FM Pulse Compression, RADAR Target Identification	130
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL